Konzeption und Entwicklung eines sprachgesteuerten Smart Home Systems unter Betrachtung des Internet of Things

Master-Thesis

für die Prüfung

Master of Science (M. Sc.)

der

Fachhochschule Dortmund/Technische Hochschule Köln

Daniel Vieth

Wiedehagen 38

48163 Münster

Matr. Nr.: 11108484

Erstprüferin:	Prof. Dr. Heide Faeskorn-Woyke
Zweitprüferin:	Prof. Dr. Birgit Bertelsmeier
Abgabedatum:	22.05.2018

I Inhaltsverzeichnis

II Abbildungsverzeichnis

III Abkürzungsverzeichnis

Begriff	Beschreibung
6LoWPAN	IPv6 over Low Power Wireless Area Networks
ADB	Andorid Debug Bridge
API	Application Programming Interface
ASR	Automatic Speech Recognition
BLE	Bluetooth Low Energy
CIoT	Consumer Internet of Things
CoAP	Constrained Application Protocol
DARPA	Defense Advanced Research Projects Agency
DBMS	Database Management System
DNN	Deep Neural Network
GMM	Gaussian Mixture Model
GPIO	General Purpose Input/Output
gRPC	(google) Remote Procedure Call
GUI	Graphical User Interface
HMM	Hidden Markov Model
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
I2C	Inter-Integrated Circuit
IAM	Indentity and Access Management
IDE	Integrated Development Environment
IIoT	Industrial Internet of Things
IoT	Internet of Things
JSON	JavaScript Object Notation
LAN	Local Area Network
M2M	Machine-To-Machine
MQTT	Message Queue Telemetry Transport
NFC	Nearfieldcommunication
NLP	Natural Language Processing

NoSQL	Not only SQL
OAuth	Open Authorization
OS	Operating System
PIoT	Public Internet of Things
Pub/Sub	Publish/Subscribe
RAM	Random Access Memory
RDBMS	Relational Database Management System
RDF	Resource Description Framework
REST	Representational State Transfer
RFID	Radio Frequency Identification
RTOS	RealTime Operating System
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol And RDF Query Language
SQL	Structured Query Language
UUID	Universally Unique Identifier
UX	User Experience
VUI	Voice User Interface
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
XML	Extensible Markup Language

1 Einleitung

1.1 Über den Autor

Die Fiducia & GAD IT AG ist IT-Dienstleister, Rechenzentrum und Softwarehaus für alle 1.000 Volksbanken und Raiffeisenbanken im deutschsprachigen Raum sowie zahlreiche Privatbanken. Der Verwaltungssitz des Unternehmens ist in Karlsruhe und Münster. Die Unternehmensgruppe Fiducia & GAD IT AG beschäftigt fast 6.400 Mitarbeiter, betreut mehr als 173.000 Bankarbeitsplätze und verwaltet knapp 83 Millionen Konten ihrer Kunden. Die Kernbankenverfahren „agree21“ und „bank21“, welche die gesamten Bankfunktionalitäten vom Zahlungsverkehr bis zum Aktiv- und Passivgeschäft abdecken, sind die aktuell produzierten Verfahren des Unternehmens.[1]

Der Verfasser dieser Arbeit ist seit 2010 als Anwendungsentwickler im Unternehmen beschäftigt. Von 2010 bis 2013 als Dualer Student der Wirtschaftsinformatik an der Hochschule Weserbergland. Die ausgeführten Tätigkeiten sind über den gesamten Zeitraum im Bereich der querschnittlichen Basisfunktionen anzusiedeln. Der Autor ist aktuell als Berater des Oberflächenframeworks sowie in der Umsetzung eines fachlichen Frameworks zur Erstellung und Individualisierung von kundenorientierten Bankprozessen beschäftigt. Seit 2014 ist der Autor Student des Verbundstudiengangs Master Wirtschaftsinformatik an der Fachhochschule Dortmund und Technischen Hochschule Köln.

1.2 Motivation und Zielsetzung

Das Internet of Things (IoT) ist aktuell eines der trendgebenden Themen der Informationstechnik. Dem Thema werden dabei viele Versprechungen, aber auch Erwartungen auf staatlicher, wirtschaftlicher und alltäglicher Ebene zugeordnet. Dabei besteht die Frage, was ist Hype und was ist Realität? Motivation und Zielsetzung dieser Arbeit ist es, dieses Thema zu untersuchen und Aspekte daraus in einem Smart Home System umzusetzen.

---

[1] Vgl. Fiducia & GAD IT AG o. V. (2018).

Durch die Vorgabe der Entwicklung eines Smart Home Systems soll jedoch nicht die Betrachtung des Themengebiets im Allgemeinen außer Acht gelassen werden. Dafür soll eine Untersuchung der aktuellen Anwendungsgebiete ebenso durchgeführt werden, wie die Feststellung der wichtigsten Technologien und des Status quo. Das Internet of Things nimmt Einfluss auf Abläufe in unserem Alltag, indem Gegenstände mit Intelligenz ausgestattet und vernetzt werden. Dabei entstehen große Datenmengen. Die Erfassung und Behandlung dieser Daten soll sowohl auf technologischer als auch auf Seiten des Datenschutzes betrachtet werden.

Der wirtschaftlich wachsende Markt des IoT, erhält zudem Einzug in Wohnungen und Häuser der Anwender. Intelligente Wohnsysteme stehen dabei noch am Anfang ihrer Verbreitung. Bekannteste Vertreter aus diesem Umfeld sind Amazons' Alexa und Google Home. Diese können auch als grundlegende Orientierung, des in dieser Arbeit zu erstellenden Systems, gesehen werden. Die Konzeption und Umsetzung eines vollständigen Smart Home Systems, kann jedoch im Umfang dieser Arbeit kein Ziel darstellen. Die Zielsetzung ist, ein System zu konzipieren und zu entwickeln, welches Basisfunktionalitäten besitzt und zukünftig eine Plattform für erweiterte Funktionen bietet. Zu diesen Basisfunktionen gehört die Verarbeitung natürlicher Spracheingaben. Die Neu- und Weiterentwicklung von Schnittstellen zwischen Mensch und Maschine, hat in den letzten Jahren, nicht zuletzt durch die Entwicklungen des IoT und den Einsatz von Geräten ohne visuelle Schnittstelle, an Bedeutung gewonnen. Systeme basierend auf Spracheingaben, haben sich in diesem Zuge, in der Performanz und vor allem der Genauigkeit, deutlich verbessert.

Um die Zielsetzung noch zu konkretisieren, weitere Aspekte, die bei der Erstellung dieser Arbeit berücksichtigt werden sollen. Klare Zielsetzung ist die lauffähige Version eines Prototyps zu produzieren. Dieser soll den technischen Durchstich verschiedener Werkzeuge, Technologien und Services, als integratives Ziel dieser Arbeit, durchführen. Eine Verknüpfung mit realen Anwendungsfällen ist zur besseren Veranschaulichung wünschenswert, aber für die Zielerreichung nicht zwangsläufig erforderlich. Gleichzeitig soll der Prototyp ein stabiles Fundament für weitere Entwicklungen auf dessen Basis bereitstellen. Der Einsatz von Technologien im Beta-Stadium ist dadurch aber nicht ausgeschlossen, wie der Einsatz des aktuell noch in der Entwicklervorschau befindlichen Android Things zeigt.

Zur Abgrenzung des Leistungsumfangs ist festzuhalten, dass eine vollständige IoT-Plattform mit Benutzer- und Gerätemanagement, konkretem Datenbankdesign sowie angeschlossenen Analyseprozessen, kein Ergebnis dieser Arbeit sein soll. Auch eine Integration von Drittanbietern, wie die bereits erwähnten Amazon Alexa und Google Home, ist kein Thema dieser Arbeit. Aufgrund von Beschränkungen der Hardwareressourcen, ist auch eine konkrete Anbindung z.B. an ein intelligentes Beleuchtungs- oder Heizungssystem eines bestimmten Herstellers nicht im Umfang berücksichtigt.

Zusammenfassend ist die Motivation und Zielsetzung dieser Arbeit, die Erstellung eines Smart Home Systems auf experimenteller, technisch orientierter Grundlage zu realisieren. Dabei steht der Aufbau von Know How, nicht nur in technischer sondern auch in theoretischer Sicht, durch die Erarbeitung von Grundlagen des IoT im Fokus.

1.3 Vorgehensweise

Zu Beginn der Arbeit wird die geschichtliche Entwicklung des Internets, von den ersten vernetzten Computern, bis zum heutigen globalen Netzwerk beschrieben. Daraufhin wird der Themenbereich des Internet of Things genauer vorgestellt. Dazu zählt die Abgrenzung des Begriffs allgemein, die Vorstellung der Treiber sowie die Einordnung in verschiedene Einsatzgebiete, von staatlichen Aufgaben bis zu Alltagssituationen. Abschließend wird das Thema Datenschutz und Datensicherheit in diesem Themenkomplex beleuchtet und Bedrohungsszenarien sowie mögliche Maßnahmen auf abstrakter Ebene dargestellt.

Im nächsten Kapitel werden die technischen Grundlagen und die durch das IoT getriebenen Entwicklungen, genauer untersucht. Es wird dargestellt, welche Technologien zum Einsatz kommen und wo sie ihre Vor- und Nachteile haben. Diese Ausführungen sind Basis des im Praxisteil umgesetzten Smart Home Systems.

Der Übergang vom letzten Abschnitt der Technologien, hin zum Themengebiet Spracherkennung, ist fließend. Das Thema Benutzerschnittstellen ist für das Internet der Dinge ein zentrales. Neue Interaktionswege mit dem Anwender müssen bereitgestellt werden, um den wachsenden Anforderungen und technologischen Weiterentwicklungen gerecht zu werden. In diesem Kapitel wird die Funktion sogenannter natürlichsprachlicher Dialogsysteme untersucht. Neben der Beschreibung der Funktionalität spielen außerdem die Vor- und Nachteile von On- und Offline-Systemen

eine große Rolle. Auch dieser Teil der Arbeit bildet ein wichtiges theoretisches Fundament für die Konzeption und Umsetzung des Prototyps. Der Spracherkennung als zentraler Funktion, wird eine hohe Bedeutung beigemessen.

Nachdem die zuvor aufgeführten Darstellungen eher ein theoretisches Fundament bilden, setzt die Konzeption bei konkreten Technologien, Funktionen und Anwendungsfällen, für das in der Zielsetzung spezifizierte Smart Home System an. Die genutzten Werkzeuge und Dienstleistungen werden vorgestellt und in den Gesamtzusammenhang des Systems eingeordnet. Die prototypische Entwicklung im anschließenden Kapitel, befasst sich mit der Umsetzung, Modellierung und Implementierung des Systems. Die im Konzept aufgestellten Funktionen und Anwendungsfälle werden anhand von Quellcodeausschnitten beschrieben und erläutert. Dabei wird auch auf besondere Eigenschaften des Betriebssystems Android Things eingegangen.

Der Ausblick erfolgt im Kontext dieser Arbeit nicht als letztes Kapitel, da hier vor allem offene Aspekte der Umsetzung aufgegriffen werden sollen. Es wird kurz aufgezeigt, welche Schritte unternommen werden müssen, um den Prototyp weiterzuentwickeln und auf eine mögliche, produktive Reife zu heben. Den Abschluss bildet die Zusammenfassung der dargestellten Themen, mit einer Bewertung der im Zuge der Arbeit beschriebenen IoT-Themen allgemein sowie besonderer Aspekte der Konzeption und Umsetzung.

2 Grundlagen des Internet of Things

2.1 Geschichtliche Entwicklung des Internets

2.1.1 Ursprünge und Web 1.0

Das Internet ist heute ein wichtiger Faktor für Wirtschaft und Gesellschaft. Es ermöglicht die Kommunikation zwischen Menschen und Maschinen, den Austausch von Daten sowie das Sammeln von Informationen - grundsätzlich - ohne an örtliche Grenzen gebunden zu sein. Die Geschichte von der ersten dokumentierten Idee eines globalen Netzwerks von Computern, von J. C. R. Licklider im August 1962[2], bis hin zu den 3,578 Milliarden Internet Nutzern weltweit im Jahre 2017[3], soll in den folgenden Abschnitten kurz beleuchtet werden.

Das ARPANET, als erstes Netzwerk von Computern, wurde von der "Defense Advanced Research Projects Agency" (DARPA), einer Behörde des Verteidigungsministeriums der USA, entwickelt. Aus der Zusammenarbeit mit verschiedenen Forschungseinrichtungen ging bspw. die Idee der Paketvermittlung von Daten zwischen einzelnen Knoten des Netzwerks aus, die Urform der bis heute bekannten TCP/IP-Protokollfamilie. Im Jahre 1969 wurden die ersten vier Knoten an verschiedenen Universtäten und deren Forschungseinrichtungen verbunden und Daten übertragen. In den folgenden Jahren gab es verschiedene weitere Netzwerke und Technologien, die vor allem auch ein frei zugängliches Netz zum Wissensaustausch zwischen Forschungseinrichtungen schaffen sollten.[4]

Der Urvater des "World Wide Web", oder historisch betrachtet dem Web 1.0, ist Tim Berners-Lee. Technologische Grundpfeiler sind die HyperText Markup Language (HTML) und das HyperText Transfer Protocol (HTTP).[5] Berners Lee erstellte im Jahre 1989 als Mitarbeiter des europäischen Kernforschungsinstitus CERN dazu einen dokumentierten Vorschlag. Die erste Webseite wurde 1990 im CERN online bereitgestellt und enthält Informationen über den grundlegenden Aufbau des Webs sowie Dokumentationen, wie andere Nutzer einen Web-Server bereitstellen können.

---

[2] Vgl. Leiner et al. (1997).
[3] Vgl. Statista (2018).
[4] Vgl. Leiner et al. (1997).
[5] Vgl. Berners-Lee (1998).

Das CERN gab die zugehörige Software im April 1993 mit der Public Domain frei und eröffnete so die weltweite Nutzung sowohl für akademische, aber zunehmend auch wirtschaftliche und private Zwecke.[6]

2.1.2 Web 2.0

Im Web 1.0 waren ein Expertenwissen und der entsprechende Zugang zum Internet nötig. Aktiv waren hauptsächlich Nutzer an Universitäten oder Privatleute, die neben dem technischen Wissen, über eine entsprechende Leitung im Telefonnetz sowie ein Modem verfügten. Dementsprechend klein war der Kreis der Personen, die aktiv das Web gestaltet haben.[7] Das änderte sich mit dem Web 2.0.

Eine einheitliche Definition was das Web 2.0 ist und vor allem wie es ausgestaltet ist gibt es nicht. Berners-Lee sieht das Web 2.0 nicht als Weiterentwicklung seiner ursprünglichen Idee an, sondern lediglich die konsequente Fortführung der Ideen aus dem Web 1.0 - dem Verbinden von Menschen, nicht das reine Verknüpfen von Computern in einem Netzwerk.[8] Diese Idee, der sozialen Interaktion im Web, steht laut Berners-Lee der technologischen Weiterentwicklung voran.[9]

Tim O'Reilly hingegen, sieht eine Weiterentwicklung des Internets hin zum Web 2.0 und hat den Begriff im Jahre 2005 maßgeblich geprägt.[10] In einem dazu veröffentlichten Artikel werden verschiedene Beispiele angeführt, die die Wandlung und Tendenzen des Internets verdeutlichen sollen. Beispielhaft ist hier Google mit seinem Dienst "Gmail" als Web 2.0 Anwendung zu nennen, die unabhängig von der Hardware nutzbar ist. Im Gegenteil dazu, gilt Netscape als klassische Web 1.0 Anwendung. Durch die zentrale Positionierung des Netscape-Browsers als De-facto-Standard für das Web, sollte Kontrolle hinsichtlich des Inhalts, der Anwendungen und vor allem hochpreisiger Server-Produkte am Markt geschaffen werden.[11]
Folgende sieben Merkmale hat O'Reilly zur Kategorisierung des Web 2.0 in seinem Artikel dokumentiert.

---

[6] Vgl. CERN (2013) sowie vgl. Berners-Lee (1998).
[7] Vgl. Adelfinger/Hänisch (2015), S. 13.
[8] Vgl. Roth (2006).
[9] Vgl. Stankovic/Jovanovic (2010), S. 3f.
[10] Vgl. O'Reilly (2005), S. 1.
[11] Vgl. O'Reilly (2005), S. 1.

- Das Web als Plattform (Gmail vs. Netscape)
- Einbeziehung kollektiver Intelligenz der Nutzer (z.B. Blogs, Wikis)
- Hohe Bedeutung von Daten als "Next Intel Inside" (Datenbanksysteme im Backend der Webanwendungen als notwendiger Standard)
- Abkehr vom klassischen Software-Lifecycle (Erweiterte Kompetenzen in der Entwicklung sowie Einbeziehung der Anwender in den Test erforderlich - dauerhafter Beta-Zustand)
- Anpassung der Programmiermodelle (leichtgewichtige Web-Services bspw. durch standardisierte Datenübertragung im Extensible Markup Language Format (XML) über HTTP)
- Erweiterbarkeit des Zugriffs über den Desktop-Computer hinaus (z.B. durch Mobiltelefone)
- Möglichkeit zur Erstellung von Anwendungen mit einer ausgeprägten User Experience (UX) (bspw. Cascading Style Sheets als Darstellungsstandrad)[12]

Insgesamt ist festzuhalten, dass durch Entwicklung zum Web 2.0, die Zahl der Nutzer, die Inhalte zum Internet beitragen, sich drastisch erhöht hat und so das soziale Netz geprägt wurde. Begünstigt wurde diese Entwicklung darüber hinaus, durch technologische Fortschritte, wie dem Breitband-Internetzugang, dem Einsatz von Open-Source-Software und auch dem wachsenden Wissen bei Entwicklern und Anwendern.[13]

2.1.3 Web 3.0 und weitere Entwicklung

Rückblickend beschreibt der Oberbegriff Web 2.0 mit seinen Ausprägungen das Internet, wie es den meisten Nutzern heute bekannt ist. Da nur die aktuelle Entwicklung betrachtet werden kann, ist eine genaue Einordung in die Geschichte und die Bedeutung für zukünftige Entwicklungen schwer möglich. Es handelt sich mehr um eine Bestandsaufnahme des aktuellen Zustands der Technik und der Interaktion der Menschen und Maschinen über das Internet.[14] So kann die aktuelle Entwicklung als Web 3.0 verstanden werden. Allerdings besteht keinerlei genaue Definition. Die Gleichung Web 2.0 + Semantik Web = Web 3.0 gilt als ein Ansatz.

---

[12] Vgl. O'Reilly (2005), S. 2ff. sowie vgl. Alby (2008), S. 15f.
[13] Vgl. Alby (2008), S. 1ff.
[14] Vgl. Alby (2008), S. XV.

Der bekannteste Anstoß hierzu stammt von Tim Berners-Lee, der das semantische Web wie folgt sieht: "The Semantic Web isn't just about putting data on the web. It is about making links, so that a person or machine can explore the web of data."[15] Hintergrund ist, dass beim Web der Hypertext Dokumente, die Verbindungen zwischen diesen Dokumenten durch Links dargestellt werden. Die Webanwendungen können diese einfachen Information aber nicht interpretieren, sodass dem Nutzer bei einer Suchanfrage eine Vielzahl von Trefferdokumenten angezeigt wird. Im semantischen Web, dem Web der Daten, soll es Computern ermöglicht werden, Informationen aus der Suchanfrage zu bewerten und semantisch zu kategorisieren. Also Verknüpfungen zwischen verschiedenen Dokumenten z.B. hinsichtlich des Ortes herzustellen. Um das Web der Daten zu ermöglichen bedarf es definierter Austauschformate und Abfragemöglichkeiten. So erfolgt die Identifikation der Dokumente über den Uniform Resource Identifier (URI), der Aufruf des Anwenders über HTTP URI und die Herstellung der Beziehung zwischen den Daten mittels XML oder dem Resource Description Framework (RDF). Darüber hinaus gibt es eine Abfragesprache SPARQL Protocol And RDF Query Language (SPARQL), zur erweiterten Abfragemöglichkeit von großen RDF-Daten.[16]

Auffällig in der aktuellen Entwicklung des Internets sind auch die 81% der 61 Millionen Internetnutzer in Deutschland (2016), die mobil im Web sind.[17] Aber nicht nur Mobiltelefone und Tablets, auch Uhren, Haushaltsgeräte und andere Dinge des Alltags, haben den Schritt ins Internet vollzogen. An dieser Stelle setzt das Themengebiet des Internet of Things an, welches neben den aktuellen Entwicklungen um das Web 3.0, bereits als nächste Entwicklungsstufe des Internets eingeordnet werden kann. Im Folgenden ist das Internet of Things zentraler Bestandteil der Untersuchung und neben der aktuellen Momentaufnahme, werden auch die damit verbunden Chancen und Risiken des IoT detailliert beschreiben.

2.2 Begriffsabgrenzung Internet of Things

Das Internet of Things gilt momentan als einer der größten Trends in der IT-Branche. Dabei wird das Schlagwort IoT an unterschiedlichen Stellen verwendet, auch als

---

[15] Berners-Lee (2006).
[16] Vgl. Berners-Lee (2006) sowie vgl. Rochow (2012).
[17] Vgl Destatis (2016).

Trendgeber im Marketing, weshalb es auch keine allgemeingültige Definition gibt. IoT stellt eher einen Sammelbegriff dar, unter dem sich verschiedene Technologien vereinen lassen, die zu großen Teilen auf bekannten Paradigmen und Entwicklungen beruhen.[18] Nichtsdestotrotz soll die folgende Annäherung abgrenzen, was IoT ist und welche Entwicklungen in der IT eine Rolle spielen.

"The IoT is a widely used term for a set of technologies, systems, and design principles associated with the emerging wave of Internet-connected things that are based on the physical environment."[19] Durch IoT wird also die Grenze zwischen der physischen Welt, in der die Menschen interagieren, mit der logischen Welt von Computern und Netzwerken verschmolzen.[20] Ein Beispiel hierfür ist das Erfassen von Sensordaten, wie der Temperatur, sowie deren Speicherung, Aggregierung und Aufbereitung für den Anwender. Alltägliche Gegenstände werden mit Intelligenz ausgestattet. Entweder indem das betreffende "Ding" selbst als IoT-Gateway fungiert oder durch ein verknüpftes Programm, welches diese Funktionen via Kommunikationsschnittstelle (z.B. WLAN) übernimmt.[21] Anwendungsfälle sind in verschiedenen Bereichen anzutreffen. Von der privaten Anwendung im Rahmen intelligenter Haushaltsgeräte (z.B. Licht- und Heizungssteuerung) oder Wearables, über die Industrie bei der Optimierung von Maschinenauslastung durch Sensoren oder der (Lager-Logistik), bis hin zu behördlichen Anwendungsfällen, wie einer intelligenten Verkehrssteuerung durch Ampeln in Smart Cities.[22] Dabei grenzt sich das Internet of Things in sofern von reiner Maschine-zu-Maschine (M2M) Kommunikation ab, indem die Kommunikation über Standards und Protokolle des Internets abläuft und diese so nutzt, wie Menschen bereits heute mit dem Web interagieren.[23]

Unter Betrachtung der zuvor erstellten Annäherung an das Themengebiet IoT und zur Verdeutlichung, dass sich verschiedene Entwicklungen der IT darunter zusammenfassen lassen, der Gartner Hype Cycle aus dem Jahre 2017 (vgl. Abbildung 1).

---

[18] Vgl. Serpanos/Wolf (2018), S. 1.
[19] Höller/et al. (2014), S. 14.
[20] Vgl. Müller (2016), S. 12.
[21] Vgl. Serpanos/Wolf (2018), S. 1f.
[22] Höller/et al. (2014), S. 14ff.
[23] Höller/et al. (2014), S. 14.

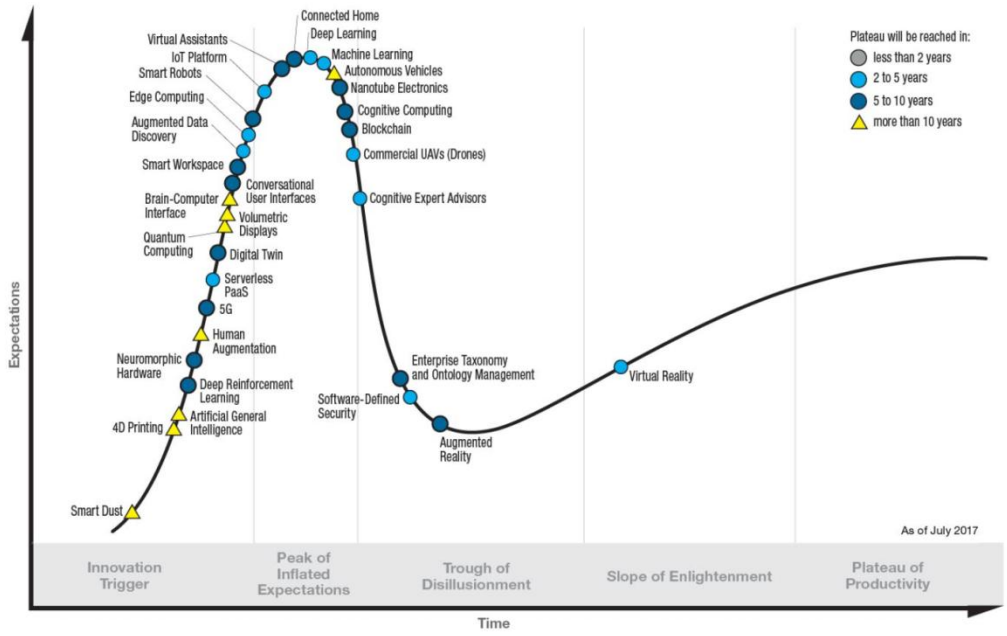

Abbildung 1: Gartner Hype Cycle for Emerging Technologies, 2017[24]

Hier wird nochmals deutlich, dass sich viele Punkte des Hype Cycles direkt oder indirekt mit dem Sammelbegriff des Internet of Things verknüpfen lassen. Ein direkter Zusammenhang besteht bei den Punkten "IoT Platform" und "Connected Home". Indirekte Bezüge lassen sich herstellen für "Conversational User Interfaces", "Virtual Assistants", "Machine Learning", aber auch "5G", zur Sicherstellung der störungsfreien Kommunikation mit dem Web, bei einer immer größer werdenden Anzahl an internetfähigen Geräten bzw. Dingen. Der Hype Cycle zeigt zudem, dass die Entwicklung des IoT und seiner Technologien bezüglich der Erwartungshaltung sich weiter aufbauen. So ist der Höhepunkt (je nach technologischem Trend) erst in den nächsten zwei bis fünf bzw. bis zu zehn Jahren zu erwarten.

Der ursprüngliche Begriff eines Internet of Things ist jedoch nicht neu und wurde schon 1999 von Kevin Ashton in einer Präsentation bei Procter & Gamble genutzt, um die neue Technologie der Radio Frequency Identification (RFID; vgl. auch Kapitel 3.2)[25] dort einzuführen. Ashton sah durch diese Technologie die Möglichkeit, dass Computer Daten aus der realen Welt überwachen, identifizieren und verstehen

[24] Gartner (2017a).

können. Ohne bei der Erfassung dieser Daten von der Eingabe durch einen Menschen abhängig zu sein.[26]

Da das Spektrum der Anwendungsbereiche so weit reicht, wird IoT auch häufig mit der Analogie "Internet of Everything" versehen.[27] Vor allem Cisco verwendet diesen Begriff, um die Vorzüge der übergreifenden Vernetzung von Personen, Prozessen, Daten und Dingen hervorzuheben.[28] Da es, wie bereits verdeutlicht, keine Definition des Begriffs Internet of Things gibt, wird nachfolgend der Begriff IoT als Sammelwort verwendet.

2.3 Entwicklung und Treiber des IoT

2.3.1 Sensoren und Mikrocontroller

Für die Entwicklung des Internet of Things bis jetzt und zukünftige Fortschritte gibt es verschiedene Treiber, die dies ermöglichen.

Als erstes sind Sensoren und Mikrocontroller zu nennen, die immer kleiner, dabei leistungsfähiger und energiesparender werden. Durch diesen Trend scheint es beim Einbau von Sensoren in unserer Umwelt kaum Grenzen zu geben. Durch die zum Teil unsichtbare Integration in Alltagsgegenstände werden diese befähigt Daten zu erfassen, diese weiterzugeben und zur Informationsgewinnung beizutragen.[29]

Moderne Mikrocontroller umfassen neben dem Prozessor, dem Random Access Memory (RAM) und einem Flash-Speicher, notwendige Netzwerk-Schnittstellen für die Ein- und Ausgabe von Daten. Die Größe ist beschränkt auf wenige Millimeter und auch der Energieverbrauch ist im Milli- bis Mikro-Watt Bereich angesiedelt. Damit die Übertragung der Daten überall gewährleistet ist, gibt es verschiedene Möglichkeiten zur Netzkommunikation (siehe Kapitel 3.2).[30]

Des Weiteren sind es nicht nur die physikalischen Gesichtspunkte, sondern vor allem auch die monetären, die als Treiber gelten. Um die Idee der vernetzten Kommunikation in alltäglichen Objekten sinnvoll einzusetzen, müssen Sensoren und Mikrocon-

[26] Vgl. Ashton (2009).
[27] Vgl. Lea (2018), S. 7 sowie vgl. Kranz (2017), S. 12.
[28] Vgl. Cisco (o. J.).
[29] Vgl. Mattern (2005), S. 39f.
[30] Vgl. Höller/et al. (2014), S. 22f.

troller in großer Anzahl hergestellt werden.[31] Neben dem Anschaffungspreis dieser Komponenten, der bei vielen Produkten zu einer vernachlässigbaren Größe geworden ist[32], sind es die Total Cost of Ownership, die hauptsächlich an den Energieverbrauch gekoppelt sind. Betrieben werden können diese durch klassische Energieversorgung wie Batterien, aber auch durch umweltabhängige Energieversorger wie das Umgebungslicht oder Thermoelektrizität. Die Fortschritte zur Speicherung dieser Energie tragen ebenfalls zur Kostensenkung im Betrieb bei.[33]

2.3.2 Netzwerktechnologien

Über Mikrocontroller und Sensoren werden Daten erfasst. Diese müssen anschließend transportiert werden, um Auswertungen und weitere Ereignisse anstoßen zu können. Hierfür sind vor allem drahtlose Netzwerke die entscheidenden Treiber, wie Bluetooth, WLAN und mobile Netzwerke wie 4G bzw. der sich in Entwicklung befindliche Nachfolger 5G.[34] Der Energieverbrauch ist auch hier der limitierende Faktor. Daher gibt es neue Spezifikationen von Netzwerken z.B. ZigBee, welches zu den Wireless Personal Area Networks (WPAN) zählt. Die Datenübertragungsraten dieser Netzwerktypen sind geringer als die der bekannten WLAN-Netze, jedoch ist auch der Energieverbrauch geringer.[35] Neben den beschriebenen drahtlosen Kommunikationssystemen, haben sich im IoT Umfeld RFID und die Nearfieldcommunication (NFC), für Übertragungen auf besonders kurzen Distanzen etabliert.

Nach einer Studie, sollen im Jahre 2020, vor allem auch durch das Internet of Things, aber bspw. auch durch Cloud Computing getrieben, 20 Milliarden Geräte mit dem Internet verbunden sein.[36] Das am weitesten verbreitete Internet Protokoll ist IPv4. Es ermöglicht jedoch nur die Vergabe von nur knapp 4,3 Milliarden IP-Adressen. Daher ist die Einführung und Verbreitung von IPv6 ein essentieller Treiber, für die Weiterentwicklung und Ausschöpfung des Potentials von IoT.[37] Weitere Entwicklungen sind trotzdem notwendig, da mit IPv6 zwar genug Adressen zur Verfügung

---

[31] Vgl. Vasseur/Dunkels (2010), S. 15.
[32] Vgl. Mattern (2005), S. 39f.
[33] Vgl. Serpanos/Wolf (2018), S. 17f.
[34] Vgl. Mattern/Floerkemeier (o. J.), S. 3ff.
[35] Vgl. Mattern/Floerkemeier (o. J.), S. 8f.
[36] Vgl. Gartner (2017b), S.2.
[37] Vgl. Sun (2016).

stehen, aber das Protokoll auch einen sehr großen Overhead hinsichtlich seines Paket-Headers besitzt, welcher dem knapp bemessenen Speicher sowie dem möglichst geringem Energieverbrauch der Mikrocontroller widerspricht.[38] Neben ZigBee wird auch an dem "IPv6 over Low Power Wireless Area Networks" (6LoWPAN) weiter gearbeitet und die Headergröße komprimiert.[39]

2.3.3 Softwareentwicklung

Bei der Entwicklung von Anwendungen sind leichtgewichtige Programmierschnittstellen (Application Programming Interface, API) und Datenaustauschformate wichtige Treiber. Die genutzten Paradigmen sind dabei keine Neuerungen. So werden im IoT-Umfeld Web-Services genutzt, die sich bereits zu Zeiten des Webs 2.0 etabliert haben. Zugriff auf die Web-Services erfolgt mittels SOAP (Simple Object Access Protocol) und REST (Representational State Transfer).[40] Diese sind neben ihrer allgemeinen Verbreitung vor allem dadurch gekennzeichnet, dass sie mit geringer Bandbreite sowie vergleichsweise kleinem Speicher auskommen. Als Datenaustauschformat hat sich neben XML, insbesondere die JavaScript Object Notation (JSON), als sehr leichtgewichtiges und strukturiertes Datenformat durchgesetzt. [41]

Darüber hinaus sind Open APIs sowie die Bereitstellung von Software Development Kits (SDK) ein wesentlicher Faktor. Mit Open APIs können Entwickler unabhängig voneinander arbeiten und speziell kleinere Unternehmen profitieren von festgelegten Schnittstellen zum Datenaustausch untereinander.[42] SDKs helfen zudem beim Einstieg in die Entwicklung von IoT-Anwendungen. Speziell die großen Anbieter wie z.B. Amazon und Google ermöglichen sowohl Firmen, als auch Privatanwendern, die Möglichkeit Funktionen mittels bereitgestellter SDKs für verschiedene Programmiersprachen zu entwickeln und gleichzeitig die zugehörigen Cloud-Services zur Verwaltung der Anwendungen, Daten und Geräte zu nutzen.[43]

---

[38] Vgl. Ray (2015) sowie Vgl. Serpanos/Wolf (2018), S. 4.
[39] Vgl. Mattern/Floerkemeier (o. J.), S. 12.
[40] Vgl. Mukhopadhyay/Suryadevara (2014), S 59.
[41] Vgl. Khodadadi/et. al (2016), S. 9f.
[42] Vgl. Höller/et al. (2014), S. 24f.
[43] Vgl. Amazon o. V. (o. J. a) sowie Google o. V. (o. J.).

2.3.4 Wirtschaft

Letztendlich sind es die wirtschaftlichen Treiber, die die Technologien rund um das Internet der Dinge weiterentwickeln. Eine Studie der CXP Group dazu sowie zu den Vorteilen von IoT für Industrieunternehmen in Europa zeigt, dass die wesentlichen Treiber Kostensenkungen und Optimierung der Prozesse im laufenden Betrieb sind. Die Studie zeigt außerdem, dass 51% der Unternehmen durch IoT die Chance sehen, neue Produkte, Services und Geschäftsmodelle erschließen zu können.[44]

Entsprechend dazu, zeigt eine Studie des Marktforschungsanbieters Gartner, dass auch im Bereich der Verbraucher (dem Consumer IoT, CIoT) erhebliche wirtschaftliche Potenziale zu erreichen sind und eine höhere Akzeptanz für das Thema IoT bei den Kunden zu schaffen ist. Die Ausgaben für diesen Bereich steigen prognostiziert von knapp einer Billion US-Dollar in 2018 auf knapp 1,5 Billionen US-Dollar im Jahre 2020.[45]

Abschließend liegt die aktuelle Erwartung der erzielten Umsätze aus dem Bereich der IoT-Anwendungen (inkl. zugehöriger Werkzeuge, Services, Support und Wartung) als zentraler Treiber bei einer Billion Euro im Jahre 2020.[46]

2.4 Einsatzgebiete

2.4.1 Übersicht

Wie in den vorherigen Kapiteln beschrieben gehen mit dem Schlagwort Internet of Things verschiedene Technologien, Paradigmen und damit auch Einsatzgebiete einher. Eine Übersicht aller Einsatzgebiete des IoT zu geben fällt daher schwer. Trotz alledem soll die nächste Abbildung den Umfang, der durch IoT abgebildet wird zeigen.

Die einzelnen Geräte (Devices) lassen sich verschiedenen Kategorien zuordnen z.B. Wearables dem Menschen, intelligente Thermostate Gebäuden und Temperatursensoren dem Gebiet der natürlichen Ressourcen. Damit die ermittelten Daten verschie-

---

[44] Vgl. CXP Group (2017).
[45] Vgl. Gartner (2017c).
[46] Vgl. Deloitte (2016), S. 8f.

denen Themenkomplexen zu Einsatzzwecken zur Verfügung gestellt werden können, bedarf es IoT-Anwendungen und -Plattformen.

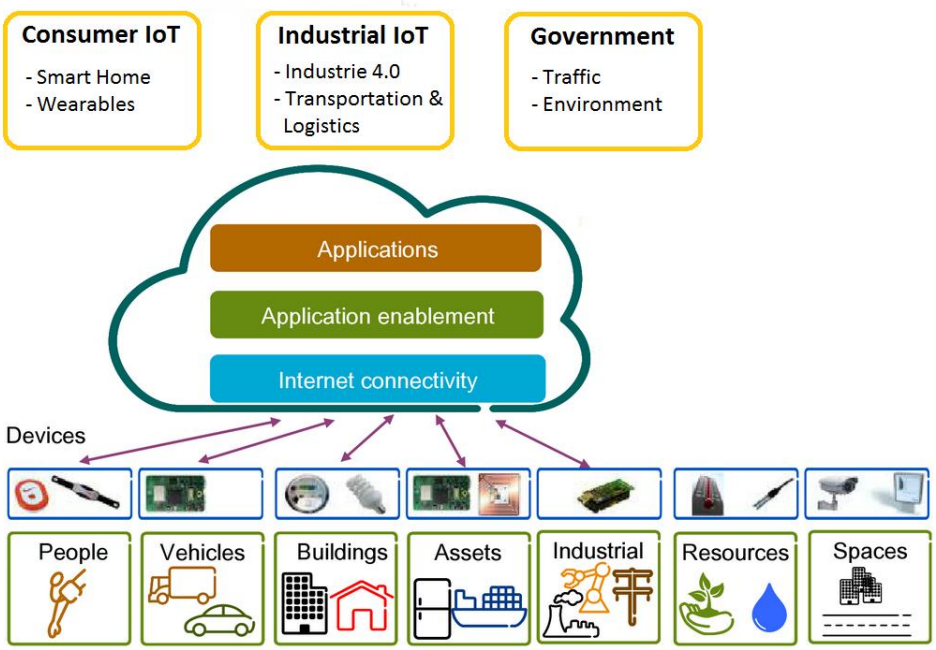

Abbildung 2: IoT-Einsatzgebiete Übersicht[47]

Die übergreifenden Themengebiete unterteilt in den öffentlichen Sektor (Government oder Public IoT, PIoT), Industrial IoT (IIoT) und Consumer IoT (CIoT), sollen anhand von Beispielen nachfolgend beschrieben werden. Einige Kategorien lassen sich dabei nicht eindeutig einem Themengebiet zuordnen. In der Abbildung nicht aufgelistet und beispielhaft hierfür, ist der Bereich Gesundheit. Dieser lässt sich grundsätzlich dem Bereich CIoT zuordnen. Jedoch auch der öffentliche Sektor, d.h. der Staat oder auch gesetzliche Krankenkassen, haben ein Interesse an den erhobenen Daten und könnten anhand dieser Rückschlüsse auf die Gesundheit der Bevölkerung schließen. Die Beispiele sind also nicht isoliert zu betrachten, sondern können auch Abhängigkeiten haben und vermitteln lediglich einen groben Überblick über den Umfang des IoT.

[47] in Anlehnung an: Höller/et al. (2014), S. 15.

2.4.2 Public IoT

2.4.2.1 Verpflichtungen

Die Herausforderung für den öffentlichen Sektor besteht darin, Regelungen und Gesetze zu schaffen, die der größer werdenden Anzahl an Anwendungen und Daten, welche durch Objekte des Internet of Things erhoben und damit auch aggregiert und analysiert werden können, gerecht zu werden. Das betrifft nicht nur Regelungen für den öffentlichen Raum z.B. autonomes Fahren, sondern auch direkt die Privatanwender. Daten die im CIoT-Umfeld durch Wearables ermittelt werden sind nicht nur für die private Nutzung der Anwender sondern auch für Firmen oder Krankenkassen sehr relevant.[48]

Besonderes soll die Mitbestimmung und Offenlegung der Datenerfassung für die Bürger als hohes Gut in die Gesetze des Datenschutzes und des Umgangs mit IoT in allen Anwendungsbereichen einfließen.[49]

2.4.2.2 Anwendungen

Viele Anwendungen im öffentlichen Sektor lassen sich unter dem Begriff Smart City vereinen. Smart Cities sind Städte die Daten zur Verbesserung des Verkehrsflusses, der Energieversorgung und Beleuchtung sowie sicherheitsrelevanten Zwecken erheben und auswerten. In Deutschland und Europa dienen diese Projekte vordergründig dem Ausbau der bestehenden Infrastruktur, während im asiatischen Raum neue Städte entstehen, die auf Basis von Sensoren und vernetzten Systemen aufgebaut sind. Diese Städte, oder ein Netz aus mehreren Städten, sollen so wertvolle Informationen bspw. für die Trinkwasserversorgung, im Zuge der immer größer werdenden Bevölkerung zu liefern.[50] Städte und Kommunen erheben heute bereits verschiedene Daten der städtischen Umwelt. Allerdings wird häufig nur ein Bruchteil dieser Daten ausgewertet, sodass der Mehrwert sowohl für die Bewohner als auch die städtische Organisation ungenutzt bleibt.[51]

---

[48] Vgl. Fraunhofer Fokus (2016), S. 5f.
[49] Vgl. Schnellbacher (2016).
[50] Vgl. Bartolo (2017).
[51] Vgl. Schnellbacher (2016).

Konkrete Anwendungen sind in der öffentlichen Sicherheit und beim Verkehr zu finden. Sensoren auf Straßen oder an Brücken können relevante Werte bzgl. Überhitzung, Kälte oder strukturellen Schäden erheben und gezielte Warnhinweise ausgeben. Zusätzlich kann durch Sensoren der Austritt von gefährliche Gasen Rettungskräfte bei ihrem Einsatz unterstützen oder die Bevölkerung frühzeitig warnen. Im Straßenverkehr überdies, sind in Zukunft verschiedene Anwendungsfälle des autonomen Fahrens denkbar, die durch Sensoren zur Verkehrsführung genutzt werden können. Hier treten wiederum die im vorherigen Absatz beschriebenen Verpflichtungen des Public IoT und der Verwaltung/Regierung ein, klare Regeln und Gesetze zu schaffen.[52]

2.4.3 Industrial IoT

2.4.3.1 Industrie 4.0

IoT wird häufig direkt verknüpft mit dem Stichwort Industrie 4.0. Mit diesem Stichwort wird die vierte Generation der Industriellen Revolution beschrieben. Nach Dampfmaschinen, Fließbandfertigung, dem Einsatz von Computern im Rahmen der Informationstechnologie, definiert die vierte Revolution die Einführung des Internets, samt vernetzter Maschinen zur besseren Auslastung von Prozessen, und die fortgeschrittene Robotertechnik.[53] IoT ist in diesem Zusammenhang als eine Komponente von Industrie 4.0 zu sehen. Viele der Technologien, welche unter den beiden Sammelbegriffen definiert sind, überscheiden sich. Dazu zählen intelligente Objekte/Maschinen, Big Data sowie selbstlernende Systeme. Die zentralen wertschöpfenden Prozesse eines Unternehmens, Entwicklung, Logistik, Produktion und Service sind davon betroffen.[54] Diese wertschöpfenden Prozesse werden durch IoT lückenlos miteinander verknüpft, sodass der Grad der Automatisierung und die M2M-Kommunikation stetig ausgebaut werden, wodurch menschliche Tätigkeiten weitestgehend ersetzet werden.[55]

---

[52] Vgl. Fraunhofer Fokus (2016), S. 9ff.
[53] Vgl. Schricks (2017), S. 9f.
[54] Vgl. Kaufmann (2015), S. 4ff.
[55] Vgl. Schricks (2017), S. 10.

2.4.3.2 Anwendungen

Anwendungen des IIoT lassen sich in vielen Industriebereichen finden. Neben der Produktionsbranche, in der Agrarwirtschaft und auch im Gesundheitssektor (Krankenhäuser). Die Ursprünge des IoT und praktische Anwendung im IIoT sind in der Warenwirtschaft und Logistik zu finden. Durch die Einführung der RFID-Tags ist es möglich eine lückenlose Verfolgung einzelner Waren oder ganzer Lieferungen jederzeit sicherzustellen.[56] Dies gilt für die gesamte Produktionskette. Warenlieferungen von Zulieferunternehmen können genauer geplant, Lagerkosten damit gesenkt und Produktionsabläufe optimiert werden.[57]

In der produzierenden Industrie können "cyber-physical Systems" (CPS), also Maschinen die mit Sensoren ausgestattet sind, den Produktionsprozess überwachen und so die Qualität des Produkts steigern. Dazu werden während des gesamten Herstellungsablaufs, Daten mittels Sensoren, z.B. Temperaturdaten, aufgezeichnet und gespeichert. Durch die stetige, automatisierte Kontrolle des zu produzierenden Produkts, können Fehlermuster in der Fertigung durch die analysierten Sensordaten frühzeitig erkannt werden. Die M2M-Kommunikation läuft automatisch, sodass direkte oder manuelle Anpassungen am Produktionsablauf ständig vorgenommen können, während wiederum weitere Daten zur Analyse des Herstellungsverfahrens aufgezeichnet werden.[58]

Darüber hinaus können neue Geschäftsmodelle zwischen Nutzer eines bestimmten Produkts und dem Hersteller geschaffen werden. Der Hersteller von Industriepumpen hat in der Vergangenheit die gesamte Pumpenanlage verkauft. Der Nutzer setzt diese ein und hat neben den Kosten des Betriebs und dessen Überwachung, auch die Kosten für die Instandhaltung und Ausfallszeiten zu zahlen. Durch Sensoren, die den Zustand des Pumpsystems stetig überwachen, kann bei einem zu erwartenden Schaden automatisch eine Handlungsempfehlung an den Hersteller ausgegeben werden. Dieser sorgt im Rahmen eines neuen Servicevertrags mit seinem Kunden für die dauerhafte Verfügbarkeit der Anlage. Der Pumpen-Nutzer hat weniger Aufwand bei

---

[56] Vgl. Fraunhofer ESK (2016), S. 9.
[57] Vgl. Kaufmann (2015), S. 25.
[58] Vgl. Kaufmann (2015), S. 27f.

der Überwachung der korrekten Funktionalität der Anlage und vor allem geringere Ausfallzeiten in seiner Produktion.[59]

2.4.4 Consumer IoT

Im Bereich des Consumer IoT stellen Wearables und Smart Home Systeme die größten Anwendungsbereiche dar. Wearables sind Systeme die im Alltag am Körper getragen werden. Sie zeichnen sich durch eine geringe Leistungsaufnahme aus und erfassen die Daten meist über unterschiedliche Sensoren zur Feststellung von Bewegungen, der Erfassung von Temperaturen oder Aufenthaltsorten. Konkrete Beispiele sind intelligente Armbänder, die mittels Sensoren die Bewegungen, die Herzfrequenz und weitere körperliche Parameter des Trägers messen. Die Daten werden entweder mit einem integrierten Display oder in einer zugehörigen Web-Anwendung bzw. App auf dem Smartphone angezeigt. Neben vielen weiteren Anwendungsfällen (Smart Watches, Smart Glasses, etc.) ist der Anwendungsbereich der persönlichen Gesundheit zu nennen.[60]

Smart Home beschreibt die intelligente Vernetzung von verschiedenen Haushaltssystemen untereinander. Zentrale Ziele eines Smart Home Systems sind die Steigerung der Wohnqualität und Sicherheit sowie die Senkung des Energieverbrauchs.[61] Dafür werden verschiedene netzwerkfähige Haushaltsgeräte miteinander verknüpft. Ein beispielhafter Anwendungsfall ist die zentrale Steuerung des Lichts über ein dediziertes Smart Home System oder einer damit verbunden Anwendung bspw. auf dem Smartphone. Die Systeme wie Licht- und Heizungssteuerung, das Öffnen und Schließen von Fenstern oder der Zugriff auf ein Kamerasystem kann auch über das Internet außerhalb des Heimnetzwerks gesteuert werden. Verschiedene Anbieter stellen Modulare- oder Komplettsysteme bereit, die als IoT-Gateway dienen und unterschiedlichen Geräte miteinander verknüpfen können. Häufig benötigt man dabei die speziellen Module sowie zugehörige Software des Anbieters, da sich diese aufgrund von Inkompatibilitäten, z.B. auf Ebene der Protokolle, nicht verbinden lassen.[62]

---

[59] Vgl. Kaufmann (2015), S. 21f.
[60] Vgl. IT Wissen o.V. (2016).
[61] Vgl. Bendel (2018).
[62] Vgl. Wendel (2017).

Ein weiterer Markt für das intelligente Zuhause entwickelt sich aktuell im Bereich der sprachgesteuerten Systeme. Diese ermöglichen häufig eine Integration der verschiedenen Anbieter und kreieren gleichzeitig für den Anwender eine neue Nutzungserfahrung. Sie arbeiten folglich als virtuelle Plattform, für welche die unterschiedlichen Anbieter eine Integration ihrer Module anbieten können. Zu den aktuell bekanntesten Systemen dieser Art zählen Amazon Alexa, Google Home und das Apple HomeKit. Die Möglichkeiten eines solchen Smart Home Systems können sehr weit gedacht werden. Ebenfalls in der Entwicklung, aber noch nicht weit verbreitet, sind intelligente Kühlschränke, die mittels Sensoren ermitteln können welche Lebensmittel fehlen und automatisiert Bestellungen aufgeben.

Besondere Herausforderung des CIoT sind mit dem Datenschutz und der sicherheitsrelevanten Aspekten der Anwender verbunden. Die Zahl der Geräte, die anwenderspezifische Daten aufzeichnen, wird immer größer.[63] Folglich sind die persönlichen, zum Teil medizinisch relevanten Daten, besonders schützenswert. Ein Zugriff von außen, sei es durch Hacker-Angriffe oder auch Unternehmen, muss so weit eingeschränkt werden, wie es die persönliche Privatsphäre und die Gesetze erlauben.

2.5 Datenschutz & Datensicherheit

2.5.1 Bedrohungsszenarien für das IoT

2.5.1.1 Allgemeine Bedrohungen

Die Bedrohungsszenarien für das IoT entstehen zum Großteil aus bekannten Szenarien der Informationstechnik. Allerdings erfahren diese durch die Digitalisierung, Vernetzung und Datenerhebung von Alltagsgegenständen, die bisher keinerlei Intelligenz besaßen, eine neue Dimension. Im Rahmen der IoT-Entwickler Umfrage der Eclipse Foundation, gibt die Mehrheit der Befragten an, beim Thema IoT und Sicherheit die meisten Zweifel zu haben.[64] Ziel dieser Bedrohungsszenarien ist der Angriff auf das Kernelement des Internet of Things, die Daten. Dazu zählt das Aufzeichnen und Manipulieren sowie die Störung der Datenübertragung.[65]

---

[63] Vgl. i-Scoop o. V. (o. J.).
[64] Vgl. Cabe (2018).
[65] Vgl. Chellapan/Sivalingam (2016), S. 188.

Herunter gebrochen auf die verschiedenen Teile, die eine IoT-Anwendung auszeichnen sind die folgenden Angriffsvektoren zu betrachten. Auf kleinster Ebene sind es die Datensammler, also die Sensoren und kleinsten IoT-Geräte. Sind diese Geräte einfach zugänglich, kann die gesamte Datenerhebung leicht manipuliert werden z.B. durch physische Beeinflussung eines Temperatursensors. Nicht alle Sensordaten sind so leicht veränderbar wie die Temperatur, aber ein Bedrohungsszenario setzt bereits hier an. Des Weiteren gelten die IoT-Gateways oder -Server als Datensammler und Datenverteiler als Angriffspunkt. Der Zugriff auf diesen Knoten des Anwendungsstapels ist durch physische Prozesse nicht möglich, jedoch sollte das Gerät über einen Authentifizierungsschutz verfügen, sodass potenzielle Angreifer keine Möglichkeit haben das Routing der Daten oder die installierte Software zu manipulieren. Der Zugriff auf die Daten durch den Anwender erfolgt häufig über externe Geräte, wie einer Webanwendung auf dem Computer oder via Smartphone. Bedrohungsszenarien dieses Angriffsvektors entstehen durch Drittsoftware, welche mögliche Malware-Infektionen des jeweiligen Endgeräts ermöglichen. Zuletzt sind es die Anwendungsserver, die durch physischen Zugriff oder durch einen Hackerangriff, ein Bedrohungsszenario hervorrufen. Neben privaten Servern von Nutzern oder Firmen, werden häufig Cloud-Dienste eingesetzt. Die Hoheit über die Datensicherheit und den Zugriffsschutz geht dadurch vom Anwender zum Dienstleister über.[66]

Für alle beschriebenen Angriffsvektoren gilt, dass die Kommunikation über physische Medien und Datenprotokolle vor Zugriffen, wie einem "man-in-the-middle"-Angriff, geschützt werden muss.

Die Auswirkungen eines Angriffs auf die dargestellten Vektoren, kann im kleinsten Kreis auf persönlicher Ebene im CIoT, im industriellen Umfeld IIoT, aber auch im öffentlichen Sektor und auf politischer Ebene stattfinden. In Wahlkämpfen werden bereits heute Prozesse der Datenanalyse eingesetzt, um Stimmungsbilder in der Bevölkerung zu erheben. Eine Verknüpfung vorhandener Daten im Zusammenspiel mit Profilen aus sozialen Netzwerken sowie Bewegungs- und Verhaltensdaten, aufge-

[66] Vgl. Brauchli/Li (2016), S. 96ff sowie Bell (2016), S. 25ff.

zeichnet durch Geräte des IoT, lassen gezieltere Initiativen, z.B. der Wahlwerbung, bezogen auf eine Bevölkerungskreis oder eine Region zu.[67]

Eine grundsätzliche Gefahr und damit einen Widerspruch gegenüber den vielen bereits beschriebenen Vorteilen des IoT zu sehen ist falsch. Neben den physischen und technischen Möglichkeiten zur Sicherstellung des Datenschutzes und der Datensicherheit, auf Basis der dargestellten Angriffsvektoren, ist es Aufgabe der Politik, Gesetze und Regelungen in Einklang mit der technologischen Weiterentwicklung zu bringen (z.B. wie ist der gesetzliche Umgang mit Nutzerdaten in der Cloud). Pro aktives reagieren durch die Politik ist häufig nicht möglich, da sich die Technologien am Markt schneller etablieren, als Gesetze geschaffen werden können.[68]

2.5.1.2 Bedrohungen im CIoT

Viele der Bedrohungen für Anwender im CIoT werden durch die vorherigen Kapitel bereits deutlich. Konkret, bezogen auf die Anwendungen aus Abschnitt 2.4.4, sind dies der Verlust der persönlichen Privatsphäre bspw. durch Zugriff auf Gesundheitsdaten (Wearables) oder Bestimmung von Persönlichkeitsmustern (Smart Home).

Da die Entwicklung eines Smart Home Systems ein zentraler Aspekt dieser Arbeit ist, sollen im Folgenden verschiedene Fragestellungen aufgegriffen werden, die Angriffsvektoren der persönlichen Privatsphäre in diesem Umfeld betreffen. Beginnend mit konkreten Beispielszenarien. Die Smart Home Systeme Amazon Echo und Google Home konnten durch einen man-in-the-middle Angriff, bekannt unter dem Namen "BlueBorne", ohne spezifische Benutzerinteraktion, durch eine bzw. mehrere Lücken des Bluetooth-Standards attackiert werden. Die Lösung dieser Sicherheitslücken durch die Hersteller auf Betriebssystemebene hilft dabei meinst nur bedingt, da aktuell vergleichsweise selten eine Verteilung von Sicherheitsupdates, bei diesen Geräten, eine große Anzahl an potenziell betroffenen Geräten offen lässt.[69] Ein weiteres Beispiel ist die Manipulation der always-on Funktionalität sprachgesteuerter Systeme. Unter der DolphinAttack wird die Aktivierung des betreffenden Smart Home Systems mittels Wakeword ("Alexa", "Ok, Google"), durch Signale im Ultra-

[67] Vgl. Howard (2016), S. 20f.
[68] Vgl. Howard (2016), S. 13.
[69] Vgl. Khandelwal (2017).

schallbereich verstanden. Dadurch können Funktionen der einzelnen Systeme ausgeführt werden, ohne das dies für den Anwender erkenntlich wird.[70]

Ist ein Smart Home System als Datensammler und IoT-Gateway von einem Angriff betroffen, können die zur Erhöhung des Komforts eingesetzten Funktionen missbraucht werden. Dazu zählt die Steuerung von Schlössern im Haus (RFID-Chip ersetzt den Schlüssel), oder die automatische Schaltung von Lampen, als Indiz für die An- oder Abwesenheit von Bewohnern.[71] Für die Manipulation eines solchen Systems ist aber der physische Zugriff auf das Gerät selbst oder das Netzwerk notwendig.[72]

Insgesamt ist, durch die vielen Akteure die sich im gesamten Anwendungsstapel eines IoT-Systems befinden, für den Endanwender schwer abzusehen, wer Zugriff auf die persönlichen Daten erhält. Neben der Herstellerfirma, können Dritte wie Cloud-Dienstleister, Vertriebsfirmen oder Werbeunternehmen, durch Sublizenzen Zugriff auf die Daten erhalten. Je nach Staat haben auch Regierungsbehörden, wie die NSA oder der Bundesnachrichtendienst Möglichkeiten auf die Daten der Nutzer, über die produzierenden Unternehmen, heranzukommen.[73] Im Zentrum des CIoT steht also die Frage, welche persönlichen Informatonen gibt der Anwender zur Komfortgewinnung preis und wie abhängig ist er von diesen Diensten?[74]

2.5.2 Maßnahmen und Regelungen

2.5.2.1 Regulatorische Maßnahmen

In der Beziehung zwischen technologischer Weiterentwicklung und regulatorischer Maßnahmen ist die neue Datenschutz-Grundverordnung der Europäischen Union (EU-DSGVO) der aktuellste Antritt. Die bisher geltende Grundordnung aus dem Jahre 1995, stammt aus den frühen Zeiten des für die Allgemeinheit zugänglichen

[70] Vgl. Zhang/et al. (2017), S. 1.
[71] Vgl. Brauchli/Li (2016), S. 100f.
[72] Vgl. Brauchli/Li (2016), S. 98.
[73] Vgl. Howard (2016), S. 13f.
[74] Vgl. Abdmeziem/Tandjaoui (2014), S. 11f sowie S. 16.

Webs (vgl. Kapitel 2.1). Die neue Grundverordnung soll die Rechte der Privatanwender stärken und Unternehmen verpflichten diese einzuhalten.[75]

Wesentlicher Eckpunkt ist die deutliche Kenntlichmachung und schriftliche Einverständniserklärung, wenn Nutzer persönliche Daten, wie die Adresse oder Telefonnummer, zur Weiterverarbeitung preisgeben. Daneben besteht ein Recht des Nutzers auf die Löschung seiner gesamten Daten sowie zugehöriger Verknüpfungen. Besonders hervorzuheben ist die Verpflichtung der sicheren Speicherung von Daten. Das gilt vor allem dann, wenn Firmen Cloud-Dienstleistungen beziehen. Sowohl das anbietende Unternehmen, bei dem der Anwender bspw. ein IoT-Gerät erworben hat, als auch der Cloud-Anbieten, müssen nach neuer Grundverordnung arbeiten.[76]

Darüber hinaus gelten die Vorgaben "Privacy by Design" sowie "Privacy by Default" nach Art. 25 der DSGVO für die Umsetzung einer Anwendung. Diese besagen, dass durch technische Maßnahmen schon in der Entwicklung einer Anwendung (z.B. Mechanismen zur Pseudonymisierung) und durch geeignete Voreinstellungen (ohne Eingriff des Anwenders, kein Zugriff durch Dritte auf die persönlichen Daten) Vorkehrungen getroffen werden müssen.[77]

2.5.2.2 Technische Maßnahmen

Als wichtigste Mittel zur technischen Sicherstellung des Datenschutzes und der Datensicherheit im IoT geben mehr als 50% der Entwickler an, auf Sicherheit in der Kommunikationsebene zu achten. Mehr als 45% setzen Verfahren zur Verschlüsselung von Daten ein.[78]

Diese Themen sind auch in anderen Bereichen der Informationstechnologie wichtige Aspekte. Verbessert werden können diese durch eine Sicherheitsarchitektur, die bereits bei der Entwicklung der Anwendung, gemäß dem Leitartikel "Privacy by Design", berücksichtigt wird. Darüber hinaus ist die Nutzung erprobter Standards für die Datenübertragung und -verschlüsselung wichtig. Aufgrund der geringeren Rechenleistung von Geräten des IoT, ist die Weiterentwicklung der bereits im Web eingesetzten Verschlüsselungsverfahren von großer Bedeutung, da diese sich durch

[75] Vgl. Beuth/et al. (2018).
[76] Vgl. Beuth/et al. (2018) sowie vgl. Nowosak (2018).
[77] Vgl. DSGVO o. V. (o. J.).
[78] Vgl. Cabe (2018).

komplexe Rechenoperationen, hauptsächlich über die Rechenleistung auszeichnen. Beispiele hierfür sind der Advanced Encryption Standard (AES), der Secure Hash Algorithm (SHA2) und auf Transportebene, das Secure Socket Layer Protocol (SSL). Leichtgewichtige Protokolle zur Datenübertragung, wie die in Kapitel 3.2.2 genauer dargestellten Protokolle Message Queue Telemetry Transport (MQTT) oder das Constrained Application Protocol (CoAP), tragen zur Verringerung der übertragenen Datenmengen und damit der benötigten Rechenleistung bei.[79]

Darüber hinaus ist eine End-zu-End Sicherheitsarchitektur von großer Bedeutung. Diese umfasst die beschriebene Verschlüsselung der gesamten Nachrichtenübertragung sowie Nutzung standardisierter Protokolle, aber auch den bidirektionalen Informationsfluss zwischen Geräten. Das erfolgt über die Nutzung einer Public Key Infrastructure (PKI) oder über den Austausch von Token zur eindeutigen Identifizierung des Kommunikationspartners. Tokens werden auch durch die beiden offenen Authentifizierungsprotokolle OAuth und OpenID genutzt und garantieren so eine sichere Benutzeridentifikation.[80]

Die häufige Verteilung von Sicherheitsupdates ist ebenfalls technische Maßnahme zur Sicherstellung des Datenschutzes und der Datensicherheit. So genannte over-the-air Updates sind ein populäres und erfolgreiches Mittel hierfür. Nach der aktuellen IoT-Umfrage der Eclipse IoT Working Group, nutzen diese Möglichkeit bereits mehr als ein Viertel der befragten Entwickler. Durch diese Maßnahme können Sicherheitslücken, wie "BlueBorne" zeitnah, auf einer großen Anzahl an Endgeräten geschlossen werden.[81]

---

[79] Vgl. Chellapan/Sivalingam (2016), S. 183 sowie S. 187ff.
[80] Vgl. Chellapan/Sivalingam (2016), S. 183 sowie Cabe (2018).
[81] Vgl. Cabe (2018).

3 Technologien des Internet of Things

3.1 Einführung

In diesem Kapitel werden die, zum Teil schon in den Grundlagen beschriebenen, Technologien der Treiber des IoT genauer beleuchtet. Der Fokus liegt dabei auf der Entwicklung von IoT-Anwendungen mit ihrem gesamten Anwendungsumfeld von der Erfassung der Daten, über die Kommunikation, bis hin zur Analyse und Verarbeitung. Die Technologiebeschreibungen werden dabei auf einem abstrakten Level vorgenommen. Eine detaillierte Betrachtung spezieller Technologien findet im Rahmen dieses Kapitels nicht statt. Einzelne Technologien dieses Abschnitts werden jedoch im Rahmen der prototypischen Konzeption und Umsetzung, des in dieser Arbeit behandelten Smart Home Systems, genauer betrachtet und praktisch eingesetzt. Grundsätzlich gilt, dass es sowohl im Open Source als auch im kommerziellen Bereich viele Technologien und Weiterentwicklungen gibt, die zur Erstellung einer IoT-Anwendung eingesetzt werden können. Hierbei gilt der Grundsatz, dass es kein "One Platform fits all" gibt und jede IoT-Anwendung einen individuellen Anwendungsstapel besitzt.[82]

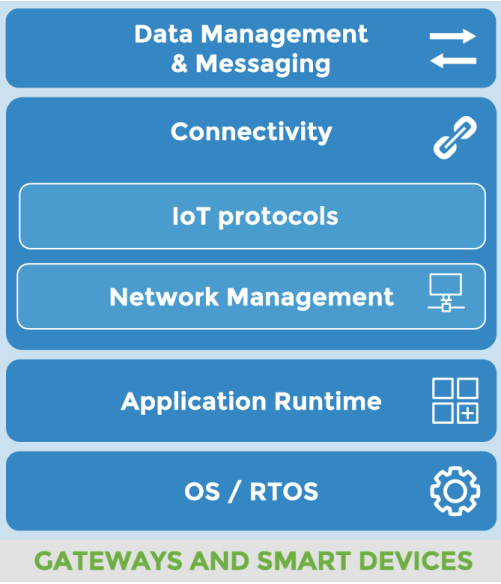

Abbildung 3: Technologien eines IoT-Anwendungsstapels[83]

[82] Vgl. Pettey (2016).
[83] in Anlehnung an: Eclipse Foundation o. V. (2018a).

Die Abbildung visualisiert den Stapel der grundlegenden Technologien, die im Rahmen einer IoT-Anwendung betrachtet werden müssen. Sie bildet die Grundlage für die untersuchten Themenschwerpunkte in diesem Kapitel und gilt übergreifend für alle Abschnitte. Die einzelnen Elemente des Stapels haben Schnittstellen zueinander, so sind die Themen "Connectivity" sowie "Data Management & Messaging" übergreifend zu betrachten. Die "Application Runtime" steht in diesem Kapitel nicht im Fokus, da die Implementierung einer IoT-Anwendung in Kapitel 6 tiefergehend betrachtet wird.

3.2 Kommunikation vernetzter Objekte

3.2.1 Datenübertragung & Netzwerke

Für die physische Übertragung von Daten verschiedener Geräte im Internet of Things, haben sich vor allem vorhandene Technologien weiterentwickelt und als Standards in diesem Umfeld etabliert. Dazu zählt das Local Area Network (LAN), in seiner kabelgebundenen und kabellosen Form des Wireless Local Area Network (WLAN), speziell im Umfeld des CIoT. Im diesem Bereich besteht meist eine entsprechende Netzwerkinfrastruktur, die für den Einsatz von IoT-Anwendungen genutzt werden kann.[84]

Wie bereits beschrieben, stammt der Ursprung des Begriffs IoT aus einer Präsentation der kabellosen Übertragungsmöglichkeit mittels RFID von Kevin Ashton. RFID ist auch heute noch ein wichtiger Standard für die Datenübertragung im Internet of Things. Ein RFID-Chip besitzt häufig keine eigene Energieversorgung (kann diese aber haben). Es handelt sich in diesem Fall um einen passiven Transponder. Energie wird durch einen aktiven Transponder mit einem Hochfrequenzsignal induziert und der Chip sendet bei Aktivierung, ein entsprechendes Signal an den aktiven Transponder zurück - bspw. eine eindeutige Schlüsselnummer. RFID-Chips sind aufgrund ihrer geringen Kosten und Größe sehr beleibt und werden auch außerhalb des IoT zur eindeutigen Identifizierung z.B. bei Etiketten, Stempelkarten oder Skipässen eingesetzt.[85] Ein weiterer Standard in der Kurzstreckendatenübertragung ist NFC. Dieser Standard ermöglicht die Kommunikation zwischen zwei NFC-kompatiblen Geräten.

---

[84] Vgl. Höller/et al. (2014), S. 103.
[85] Vgl. Mattern (2005), S. 55f.

Die Funktionsweise ähnelt der des RFID-Verfahrens, da meist einer der NFC-Transponder den aktiven Teil und der andere Chip den passiven Teil darstellt. NFC-Chips sind in aktuellen Smartphones-Standard und erzielen eine hohe Akzeptanz durch intuitive Nutzbarkeit sowie Sicherheit, da der Datenaustausch nur möglich ist, wenn beide Akteure sich in einem Abstand von wenigen Zentimetern zueinander befinden.[86]

Ebenfalls drahtlose Übertragung von Daten für Kurzstreckenkommunikation (bis zu 50m) liefert der Bluetooth-Standard. Dieser ermöglicht eine hohe Datenrate und ist ebenfalls im CIoT ein beleibter Standard. Konkrete Beispiele hierfür sind im Gesundheitswesen und bei Wearables zu finden. Diese Geräte haben häufig keine eigene Internetverbindung und übertagen ihre Daten mittels Smartphone (mobile Netze/LAN). Im IoT-Umfeld wird häufig Bluetooth Low Energy (BLE) genutzt, der einen optimierten Einsatz von Energieressourcen erzielt.[87] Des Weiteren kann Bluetooth in Kombination mit RFID/NFC eingesetzt werden. Mittels RFID/NFC wird lediglich die Geräteidentifizierung und -verknüpfung hergestellt, die weitere Übertragung der Daten erfolgt über den weitaus höhere Datenmengen verarbeitenden Bluetooth-Standard.[88]

Zur Datenübertragung von Mikrocontrollern z.B. Sensoren, die an ein IoT-Gateway direkt angeschlossen sind, gibt es Bus-Systeme. Bekanntestes Beispiel ist Inter-Integrated Circuit (I2C), welches die Daten über einen Bus austauscht und durch das Gateway mit dem Web verknüpft.[89] Unterstützt werden unterschiedliche Mikrocontroller und Betriebssysteme, zwischen denen Daten über zwei Bus Linien, Serial Data Line (SDL) und Serial Clock Line (SCL), ausgetauscht werden.[90]

3.2.2 Protokolle & Nachrichten

Die Elemente "Connectivity - IoT Protocols" und "Messaging" des IoT-Anwendungsstapels aus Abbildung 3, haben Schnittstellen zueinander und werden übergreifend betrachtet. In Kapitel 2.3 ist beschrieben, dass die Headerdaten der Pro-

---

[86] Vgl. Mattern (2005), S. 49f.
[87] Vgl. Höller/et al. (2014), S. 103.
[88] Vgl. Mattern (2005), S. 49.
[89] Vgl. Chellapan/Sivalingam (2016), S. 190.
[90] Vgl. NXP Semiconducters o. V. (2014), S. 3.

tokolle sowie deren Geschwindigkeit in der Datenübertragung und dem Energieverbrauch für IoT-Geräte entscheidend sind. Bekanntestes Paradigma der Protokolle zum Nachrichtenaustausch im Web ist HTTP mit dem Request/Response-Verfahren. Die Vorteile dieses Verfahrens sind neben der großen Bekanntheit, vor allem die weite Verbreitung.[91] Da für das Internet of Things jedoch besonders leichtgewichtige Protokolle zur Übertragung vieler Nachrichten zentral sind, gibt es weitere Protokolle und Verfahren für das Messaging, die auch im Zuge der Entwicklung des IoT entstanden sind oder weiterentwickelt wurden.

Ein bekanntes Portokoll ist Message Queue Telemetry Transport (MQTT). Spezielle Eigenschaften dieses Verfahrens sind der geringe Energie- und Bandbreitenverbrauch sowie die lose Kopplung zwischen den einzelnen Akteuren. Im Vergleich zu HTTP, wird eine um ein Vielfaches höhere Rate an Nachrichten pro Stunde erzielt, bei gleichzeitig geringerer Nutzung von Energieressourcen.[92] Mit einer maximalen Nachrichtengröße von 256 MB wird deutlich, dass sich dieses Protokoll vor allem für die häufige Übertragung von Nachrichten, mit einem kleinen Nachrichteninhalt, eignet.[93]

Erzielt wird dies, neben dem geringeren Nachrichten-Header, vor allem durch eine Nachrichtenzentrierung im Rahmen des Publish/Subscribe-Verfahrens (Pub/Sub), als Gegensatz zum dokumentorientierten HTTP Request/Response-Verfahrens. Beim Pub/Sub-Verfahren müssen sich Nachrichten-Sender und -Empfänger nicht kennen. Der Sender (Publisher), überträgt Nachrichten zu einem definierten Thema (Topic), welches durch den Nachrichtenempfänger (Subscriber) abonniert werden kann. Die Daten werden bei diesem Paradigma ins Zentrum gerückt. Während im klassischen HTTP-Umfeld Anfragen an einen Server per URL gesendet werden, abonniert der Subscriber nur Topics, die er für die weitere Verarbeitung benötigt.[94] Konkretes Beispiel ist die Erhebung von Temperaturdaten. Ein Sensor misst die aktuelle Umgebungstemperatur, überträgt diese Daten z.B. mittels I2C an das Anwendungssystem, welches als Message Broker fungiert und die ermittelten Daten via Nachricht zum Thema "Temperatur" veröffentlicht (Publish). Ein zentraler Server abonniert das

[91] Vgl. Serozhenko (2017).
[92] Vgl. Karasiewicz (2013).
[93] Vgl. Nicholas (2012) sowie vgl. Karasiewicz (2013).
[94] Vgl. Serpanos/Wolf (2018), S. 7ff sowie vgl. Roth (2016).

Thema "Temperatur" und kann auf Basis der übertragengen Daten weitere Operationen oder Verknüpfung mit anderen Topics bspw. "Luftdruck" vornehmen.

Weitere Protokolle, die im Internet der Dinge eine größere Bedeutung besitzen und an dieser Stelle notiert werden sollen, sind das Constrained Application Protocol (CoAP) und das Extensible Messaging and Presence Protocol (XMPP). Während CoAP auf das Paradigma der REST-Services (Request/Response) aufbaut und unterschiedliche Datenformate, wie JSON und XML unterstützt[95], setzt XMPP auf den XML-Standard und wird zeichnet sich vor allem durch die Echtzeit Nachrichtenübermittlung aus.[96]

3.3 Betriebssysteme

Die Anforderungen an ein Betriebssystem lassen sich aus den bisher dargestellten Anforderungen und Treibern, die das Internet der Dinge im Allgemeinen betreffen, ableiten. Zentrale Anforderungen sind die speicher- und energieeffiziente Ausführung von Anwendungen sowie die Bereitstellung von Treibern und Bibliotheken zur Nutzung von Standards. Der letzte Punkt ist speziell dahingehend wichtig, Anwendungsentwicklern die Möglichkeit zu geben, Anwendungen zu konzipieren und umzusetzen, ohne das tiefe Kenntnisse im Maschinencode oder maschinennahen Programmiersprachen notwendig sind. Neben der Anforderung an die Sicherheit, sind auch deterministische Latenzzeiten in Form von Real Time Operating Systems (RTOS) ein wichtiger Faktor.

Beispiel für ein Echtzeit Betriebssystem ist FreeRTOS. Dieses RTOS unterstützt verschiedene Mikrocontroller und zeichnet sich dadurch aus, dass auftretende Ereignisse exakt definiert und verschiedene Prozesse zeitgenau gesteuert werden können.[97] Einsatzgebiete eines solchen Betriebssystems finden sich in der Automobilbranche und Medizin, wo eine zuverlässige Echtzeitsteuerung essenziell ist.[98] Ebenfalls ein quelloffenes Open Source Betriebssystem, ohne Fokussierung auf Echtzeitsteuerung, ist das OS Contiki. Es unterstützt ebenfalls verschiedene Mikro-

[95] Vgl. Mukhopadhyay/Suryadevara (2014), S 56f sowie vgl. Bormann (o. J.).
[96] Vgl. XMPP o. V. (o. J.).
[97] Vgl. Vasseur/Dunkels (2010), S. 131f.
[98] Vgl. Kobylinska/Martins (2017).

controller und ist in C programmiert, was eine leichte Portierbarkeit ermöglicht.[99] Als erstes OS im IoT-Umfeld hat Contiki den gesamten IPv6-Stack zur Verknüpfung netzwerkfähiger Geräte unterstützt. Des Weiteren sind die Betreibssysteme TinyOS, RIOT OS und Canonical Ubuntu for IoT aufzuführen. Jedes dieser Systeme adressiert, neben den grundsätzlichen Anforderungen, weitere Bereiche, die je nach Anwendungsfall eine unterschiedliche Bedeutung besitzen.[100]

Einplatinencomputer, wie der Raspberry Pi oder der Arduino, verfügen über mehr Rechenleistung als ein einfacher Mikrocontroller. Dementsprechend haben die Hersteller weit verbreiteter, kommerzieller Betriebssysteme, wie Microsoft und Google eigene Entwicklungen für das Internet der Dinge veröffentlicht. Bei den OS Windows 10 IoT und Android Things, handelt es sich, vereinfacht dargestellt, um leichtgewichtige Versionen der bekannten Mutter-Systeme. Allerdings angereichert durch Treiber für Sensoren und spezieller weiterer Funktionen, die die Sicherheit und das Handling bei der Vergabe von Rechten betreffen. Abgegrenzt werden können diese beiden Arten in Basic OS (Contiki, FreeRTOS, u. w.) und Advanced OS (Windows IoT Core, Android Things).[101] Android Things ist auch Basis des in dieser Arbeit erstellten Prototyps und wird in Kapitel 5.3.2 genauer vorgestellt.

3.4 Datenbanken

3.4.1 Anforderungen

Im Folgenden werden die grundlegenden Anforderungen, die sich für eine Datenbank im Umfeld einer IoT-Anwendung ergeben, betrachtet. Dabei gilt wiederum der Grundsatz: Es gibt kein allgemeingültiges Rezept zur Bestimmung, der am besten geeignetsten Datenbank, oder des Datenbanktyps. Der Fokus auf bestimmte Eigenschaften eines Datenbanksystems (DBMS) ist für jede Anwendung differenziert zu bewerten. Mit der immer weiter wachsenden Anzahl an Geräten zur Datenerfassung, in der Industrie und im Alltag, ist eine "Datenexplosion" zu erwarten. Damit IoT

[99] Vgl. Vasseur/Dunkels (2010), S. 129f.
[100] Vgl. Vasseur/Dunkels (2010), S. 129ff sowie vgl. Kobylinska/Martins (2017).
[101] Vgl. Höller/et al. (2014), S. 83ff.

nicht zum Selbstzweck wird, sondern relevante Daten zu Informationen und Wissen transformiert werden können, ist die Bedeutung der Datenhaltung weitreichend.[102]

Generell gilt für IoT-Anwendungen, dass Daten häufig in unstrukturierter Form und großer Menge vorliegen. Eine Flexibilität hinsichtlich der Skalierbarkeit, des Persistenz-Schematas der Daten oder Objekte sowie eine hohe Fehlertoleranz, ist in vielen Anwendungsfällen des IoT von hoher Bedeutung.[103] So kann sich das Datenaufkommen bei CIoT-Anwendungen mit zunehmender Nutzerzahl oder erweiterten Funktionalitäten ständig ändern. Gleiches gilt für Anwendungen des IIoT und des öffentlichen Sektors. Beim Austausch von Sensoren können neue Daten entstehen, die durch das bisherige Datenmodell nicht abgebildet werden. Die effiziente und konsistente Integration in die Erhebungs- und Analyseprozesse ist hier die Herausforderung.[104] Außerdem wird die Datenmenge größer, je mehr Knoten sich in einem solchen Netzwerk von Sensoren oder IoT-Gateways befinden.

Neben Rohdaten von Sensoren, müssen heterogene Daten unterschiedlichsten Typs, textuell, grafisch oder auditiv gespeichert und zum Zweck des Wissensbildung zugreifbar gemacht werden.[105] Dafür sind Such- und Abfragesprachen notwendig, die eine Selektion oder Aggregierung von Datensätzen umsetzen. Diesen Anforderungsschwerpunkten widmet sich das im Folgenden Abschnitt behandelte Thema Big Data unter Zuhilfenahme der im Kapitel 3.4.3 beschriebenen relationalen- und NoSQL-Datenbanken.

3.4.2 Big Data

Zur Adressierung der großen Datenmengen, welche im Lebenszyklus einer IoT-Anwendung entstehen, wird der Begriff Big Data verwendet. Dieses Schlagwort findet nicht nur im Umfeld des IoT Beachtung, sondern ist aktuell einer der wichtigsten Entwicklungstrends in der Informationstechnologie. Investitionen in diesem Kontext werden in den Bereichen der Kauf- und Bestellprozesse von Online-Systeme, Zahlungsverkehrsströmen und zu Monitoring Zwecken in der Industrie eingesetzt.[106]

---

[102] Vgl. Bell (2016), S. 1ff
[103] Vgl. Marmonti (2016) sowie vgl. Sullivan/Sullivan (2016).
[104] Vgl. Meier (2018), S. 5.
[105] Vgl. Meier (2018), S. 5f.
[106] Vgl. Meier (2018), S. 5.

Da die Einsatzgebiete unter dem Schlagwort Big Data sehr weitreichend sind, liefert die Definition der Forschungs- und Beratungsfirma Gartner, einen groben Überblick über die wesentlichen Eigenschaften: "Big data is high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation."[107] Anders ausgedrückt, ist die kosteneffiziente Verarbeitung von Daten zu Wissensgewinnung und Unterstützung bei der Entscheidungsfindung in Unternehmen, nicht auf Basis altbekannter Werkzeuge der Datenbankzugriffsmechanismen durchzuführen.[108]

Die wesentlichen Eigenschaften einer Big Data-Anwendung lassen sich unter den drei V's sammeln.

- Volume: Bezeichnet die großen Datenmengen im Terra- bis Zettabereich.
- Variety: Umfasst die Vielfalt der strukturierten und unstrukturierten Daten unterschiedlicher Herkunft (Rohdaten, Dokumente, Audio, Video).
- Velocity: Charakterisiert die Geschwindigkeit in der Verarbeitung und Analyse von Daten, welche größtenteils im Sekundenbereich oder in Echtzeit geschieht.

Die drei V's, die zur allgemeinen Definition eines Big Data-Systems zählen, werden durch zwei weitere ergänzt. Das ist zum einen der Wert - "Value", welcher die Erhöhung des Unternehmenswerts, der aus den getätigten Investitionen in Hardware, Software und Personal, entsteht. Zum anderen ist es die Wahrhaftigkeit "Varacity" der Daten. Diese umschreibt den Qualitätsgrad der Daten, der sowohl von der erhobenen Menge als auch von weiteren Faktoren, wie statistischen Algorithmen und daraus folgenden Fehleinschätzungen, abhängt[109].

3.4.3 Relationale und NoSQL-Datenbanken

Zunächst sollen an dieser Stelle die grundlegenden Unterschiede zwischen relationalen- und NoSQL-Datenbanken aufgezeigt werden.

---

[107] Gartner o. V. (o. J.).
[108] Vgl. Bell (2016), S. 19 sowie vgl. Gartner o. V. (o. J.).
[109] Vgl. Meier (2018), S. 6f.

Relationale DBMS (RDBMS) basieren auf einem fest definierten Datenmodell aus Tabellen und Spalten. Für dieses Modell gibt es Normalformen, die sicherstellen, dass funktionale Abhängigkeiten von Datensätzen einer Tabelle aufgelöst werden, indem sie mittels Primär- und Fremdschlüssel, auf unterschiedliche Tabellen aufgeteilt werden. Bei Einhaltung der Normalformen, garantieren RDBMS so eine nichtredundante Datenhaltung innerhalb des Modells.[110] Die Eigenschaften einer relationalen Datenbank werden durch das ACID-Prinzip (Atomicity, Consistency, Isolation, Durability) beschrieben. Dieses sagt aus, dass die Operationen innerhalb einer Transaktion nur vollständig, oder ohne Auswirkung auf die Datenbank ausgeführt werden (Rollback). Darüber hinaus ist die Datenkonsistenz höchstes Gut eines relationalen Systems. Zeitgleich durchgeführte Transaktionen bedingen sich gegenseitig nicht, sodass ein Mehrbenutzerbetrieb, durch unterschiedliche Isolationsverfahren, möglich ist, ohne die Datenkonsistenz zu beeinflussen.[111] Zur Definition des Datenbankschematas sowie zur Selektion und Manipulation von Datensätzen innerhalb der Tabellen, wird die Structured Query Language (SQL) eingesetzt.

Relationale Datenbanksysteme sind in der Wirtschaft immer noch die führenden Systeme und zeichnen sich neben den beschriebenen Eigenschaften vor allem dadurch aus, dass das Know-How in diesem Umfeld sehr weit verbreitet ist und sie als etablierte Systeme gelten. Die prominentesten Beispiele sind im Open Source-Bereich MySQL und PostgreSQL sowie im kommerziellen Bereich Oracle, Microsoft SQL Server und DB2.[112]

Im Kontext von verteilten Systemen haben NoSQL-Datenbanken ("Not only SQL"), im Laufe der Entwicklung des Web 2.0, an großer Bedeutung gewonnen. Sie heben sich von RDBMS dadurch ab, dass sie schemafrei oder nur geringe Schemata-Restriktionen besitzen und horizontal skalierbar sind. Das bedeutet, ein verteilter Cluster von Servern kann durch weitere Knoten ergänzt werden und der Datendurchfluss wächst im selben Faktor mit. Ferner besitzen NoSQL-Datenbanken keine Abfragesprache, sondern stellen APIs für verschiedene Programmiersprachen, zur Einbettung der Datenbankenzugriffe in die Anwendung, bereit.[113] Die Schlussfolge-

---

[110] Vgl. Meier (2018), S. 9f.
[111] Vgl. Meier (2018), S. 30.
[112] Vgl. DB-Engines o. V. (2018).
[113] Vgl. Edlich/et al. (2011), S. 1f sowie S. 33f.

rung aus den aufgeführten Eigenschaften ist das CAP-Theorem von Eric Brewer. Dieses sagt aus, dass in einem verteilten System nur zwei der drei Eigenschaften (Consistency, Availability und Partition Tolerance) erfüllt sein können.[114] Um im Gegensatz zum ACID-Modell, in dem die permanente Konsistenz der Daten prägend ist, mit verteilten Datenbanksystemen eine alternatives Konsistenzmodell bereitzustellen, wurde das BASE-Prinzip (Basically Available, Soft State, Eventually Consistent) entwickelt. Dieses besagt zusammengefasst, dass NoSQL-Systeme meist verfügbar sind, auch wenn ein Knoten des Clusters ausfällt (Basically Available). Dagegen ist die Konsistenz zu jeder Zeit, im Vergleich zum ACID-Prinzip, nicht oberste Priorität. In einem NoSQL-System kann die Entscheidung zur zwischenzeitlichen Inkonsistenz der Daten bewusst getroffen werden, sodass die Konsistenz erst zu einem späteren Zeitpunkt wieder hergestellt wird (Eventually Consistant und Soft State).[115]

Einteilen lassen sich die NoSQL-Kernsysteme in Wide Column Stores, Document Stores, Key/Value Stores und Graphendatenbanken. Key/Value-Systeme bieten im einfachsten Fall ein Schlüssel und Werte-Paar als Datenmodell an. Sie zeichnen sich durch eine schnelle Datenverwaltung aus, sind jedoch was Abfragemöglichkeiten betrifft, auf die Möglichkeiten der API beschränkt. Zu den Document Stores zählen die Systeme MongoDB, CouchDB und die Firebase Realtime Database. Es wird meist nur das Format der ID zur Adressierung der Datensätze definiert, die zugehörigen Dokumente besitzen Datenformate wie JSON oder RDF. Besonderheit der Column Stores ist, dass eine beliebige Anzahl an Schlüsseln, auf eine beliebige Anzahl an Key/Value-Datensätzen, gebunden werden kann. Bekannte Vertreter sind Cassandra und HBase. Das letzte der NoSQL-Kernsysteme sind die Graphendatenbanken. Diese finden speziell dort Einsatz, wo Beziehungen zwischen einzelnen Knoten durch Kanten dargestellt werden, z.B. Freundschaftsbeziehungen in einem Sozialen Netzwerk oder Beziehungen zwischen Geodaten.[116]

Zusammenfassend bedient sich das Internet der Dinge sowohl relationale Datenbanken als auch NoSQL-Datenbanken. NoSQL-Systeme passen grundsätzlich, durch

---

[114] Vgl. Edlich/et al. (2011), S. 31ff.
[115] Vgl. Meier (2018), S. 34f.
[116] Vgl. Edlich/et al. (2011), S. 7ff.

ihre Schemafreiheit hinsichtlich unstrukturierter Daten und die leichte Skalierbarkeit im Bezug auf große Datenmengen, bei gleichbleibender Performanz, besser zu den Anforderungen einer IoT-Anwendung. Allerdings sind konsistente Datenhaltung für Kernfunktionalitäten eines Unternehmens noch ein wichtiger Faktor in der Entscheidungsfindung vieler Betriebe. Eine Verschmelzung der Datenbanktypen kann also ebenso eine praktikable Lösung darstellen.[117]

An dieser Stelle nicht vollständig unterschlagen werden sollen in-memory Datenbanken wie MemSQL (relational) oder Hazelcast (Key/Value Store). Mit diesen ist eine weitere Optimierung der Performanz in beiden Welten möglich. Die Verarbeitung im Arbeitsspeicher oder Prozessorcache ist jedoch nicht für jede IoT-Anwendung praktikabel, da vor allem einfache Mikrocontroller nur einen geringen Arbeitsspeicher besitzen.[118]

3.5 Cloud Services

Unter dem Begriff Cloud Services oder Cloud Computing sammeln sich verschiedene Dienstleistungen, die in Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), Infrastructure-as-a-Service (IaaS) und allgemein Everything-as-a-Service (XaaS), wie z.B. Datenbase-as-a-Service (DBaaS) unterteilt werden können. Eine weitere Unterteilung ist in die Public Cloud (Anbieter stellen Services über das Internet und APIs zur Verfügung) sowie die Private Cloud (Services werden in einem firmeninternen Netz zur Verfügung gestellt) sinnvoll.[119] Der Fokus in dieser Arbeit, liegt auf den Public-Clouds, die einem Kunden die in Abbildung 4 skizzierten Services zur Verfügung stellen.

Im Internet of Things sind diese Angebote der Cloud-Dienstleister aus verschiedenen Gründen von besonderem Interesse. Sie erleichtern den Einstieg in die Entwicklung von IoT-Anwendungen, da zunächst keine eigenen Hardware-Ressourcen, abgesehen vom IoT-Gerät selbst, bereitgestellt werden müssen. Dabei wird grundsätzlich der gesamte Anwendungsstapel, der in den vorherigen Kapiteln beschriebenen Technologien, abgebildet. Dazu zählt das Thema Datenbanken und Datenmanagement,

---

[117] Vgl. Scholing (2015).
[118] Vgl. Sullivan/Sullivan (2016).
[119] vgl. Metzger et al. (2011), S. 2f, 21f.

samt Skalierungs- und Replikationsfunktionen. Auch für komplexe Themen, die ein tiefgehendes Wissen des Anwendungsentwicklers voraussetzen, wie Analyseverfahren oder Algorithmen des Machine Learnings, werden durch den Anbieter geleistet ("Event Management, Analytics & UI").[120] Im Grunde sind auch Backend-Funktionalitäten zur Weiterverarbeitung für IoT-Geräte, mit geringer Rechenleistung, in der Cloud abbildbar. Dafür bieten die Dienstleister APIs unterschiedlicher Programmiersprachen zur Nutzung der einzelnen Funktionen, aber auch der Protokolle wie Pub/Sub und MQTT, an ("Application Enablement", "Connectivity", "Message Routing").[121]

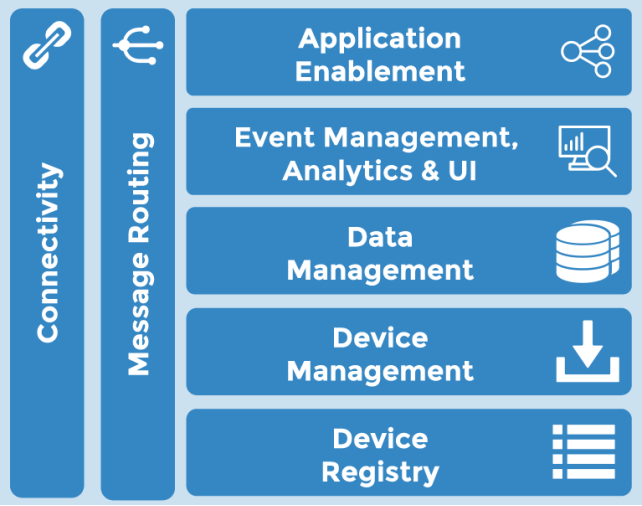

Abbildung 4: IoT-Anwendung Cloud Plattform[122]

Wie die oben stehende Abbildung ebenfalls verdeutlicht, kann das "Device Management" sowie die "Device Registry", vollständig verwaltet in der Cloud-Plattform aufgesetzt werden. Das Gerätemanagement umfasst Sicherheitsmechanismen zur Authentifizierung und Daten-Zugriffskontrolle, sowohl für die Endanwender als auch für die Entwickler der IoT-Anwendung. Die Kosten für die Nutzung der Services sind für jede Funktion verschieden und meist an Datenvolumina oder Zugriffe pro Zeiteinheit gebunden. Bei geringer Auslastung ist die Nutzung häufig kostenlos, sodass vor der Festlegung eines Dienstleisters, die angebotenen Services, mit den Anforderungen an die zu entwickelnde Anwendung, abgeglichen werden können.

---

[120] Vgl. Farzad/et. al (2016), S. 13f sowie vgl. Bial/Rock (2018).
[121] Vgl. Bial/Rock (2018).
[122] Eclipse Foundation (2018b).

Da es sich um vollständig verwaltete Services eines Anbieters handelt, gelten die Datenschutzrichtlinien und Gesetze des Staates, in dem die produzierenden Server stehen. Mittelständische Anbieter wie Q-Loud operieren nur in Deutschland, kommerzielle Anbieter wie Amazon (Amazon Web Services), Microsoft (Azure) und Google (Cloud Platform), bieten Server auf allen Kontinenten an.[123]

3.6 Benutzerschnittstellen

Die Besonderheit von Benutzerschnittstellen im Internet der Dinge ist, dass eine Abkehr von klassischen, visuellen Schnittstellen, wie sie in der Welt des Webs bisher bekannt waren, in vielen Anwendungsfällen nötig ist. Die Eingabe über Maus und Tastatur mit einer grafischen Benutzeroberfläche, oder die Bedienung über einen Touchscreen, ist bei Wearables wie Fitnessarmbändern, aber auch bekannten Smart Home Systemen (Amazon Echo, Google Home), entweder nicht möglich oder nicht vorgesehen. Daher gibt es neue Wege zur direkten (Sprachschnittstelle) oder indirekten (Begleitanwendung auf einem Smartphone) Interaktion mit Geräten des Internets der Dinge.[124]

Benutzerschnittstellen wie Gesten- oder Sprachsteuerung befinden sich, auch wenn vor allem im Bereich der Sprachsteuerung schon deutliche Fortschritte erzielt wurden, in den Kinderschuhen.[125] Dennoch ergeben sich neue Möglichkeiten in der Mensch-Computer-Interaktion und bekannte Paradigmen werden neu aufgesetzt. Im Folgenden Kapitel wird die Interaktion mit natürlichsprachlichen Dialogsystemen, im Kontext des Internet of Things, detailliert beleuchtet.

---

[123] Vgl. Bial/Rock (2018).
[124] Vgl. Mattern/Floerkemeier (o. J.), S. 3.
[125] Vgl. Mattern (2005) S. 41.

4 Spracherkennung

4.1 Einordnung

Für die Spracherkennung im Allgemeinen gibt es viele Anwendungsbereiche in denen sie eingesetzt wird. Die Wissenschaft der Computerlinguistik befasst sich mit der Erstellung von formalen Sprachmodellen, um eine syntaktisch und semantisch korrekte Ableitung der Eingaben zu ermöglichen. Systeme die Spracherkennung zur Kommunikation zwischen Mensch und Maschine einsetzen, heißen Dialogsysteme.[126] Dialogsysteme zur Verarbeitung von natürlicher Sprache sind aktuell ein großer Trend und im Alltag der Anwender angekommen. Zum Vergleich: In der kürzeren Vergangenheit hat sich der Anteil der Suchanfragen bei Google mittels Spracheingabe von weniger als einem Prozent, auf über zehn Prozent gesteigert.[127] Die Gründe hierfür werden im Abschnitt 4.2 deutlich.

Dialogsysteme lassen sich unterscheiden zwischen Systemen der reinen Informationsabfrage und transaktionalen Systemen. Die Eingaben können grundsätzlich mittels Texteigabe über eine Tastatur oder natürlichsprachlich, wie bei Sprachassistenten oder telefonbasierten Dialogsystemen erfolgen. Da in dieser Arbeit vorrangig die natürlichsprachliche Erkennung im Rahmen der Mensch-Maschine-Interaktion des CIoT im Fokus steht, werden diese Sprachsysteme nun genauer betrachtet. Der Nutzer kann Verkehrsinformationen oder aktuelle Daten seines Bankkontos (Informationssysteme) abfragen. Darüber hinaus gibt es Systeme zum Kauf von Kinotickets und der Buchung von Flugtickets (Transaktionssysteme).[128] Eine weitere Unterteilung solcher Systeme wird definiert durch "cost saving applications" (standardisierte Aufgaben z.B. eines Call-Centers, Reservierung eines Hotelzimmers) und "value add applications" (dem Anwender wird ein Mehrwert geboten z.B. in Form einer neuartigen UX, Smart Home Systeme).[129]

Bezogen auf das CIoT, handelt es sich bei diesen Dialogsystemen um "value add applications" mit der Besonderheit, dass sie sowohl Informations- als auch Transak-

---

[126] Vgl. Carstensen/Jekat/Klabunde (2010), S. 6f. sowie Thar (2015), S. 43f.
[127] Vgl. Branscombe (2017).
[128] Vgl. Thar (2015), 45f.
[129] Vgl. Thar (2015), S. 47.

tionssysteme sind und in eine gesamte Anwendungslandschaft eingebunden (vgl. Kapitel 5.4).

Die Notwendigkeit zur Entwicklung neuer Interaktionsmöglichkeiten liegt darin begründet, dass durch das Internet der Dinge, Geräte, welche keine klassische Eingabemöglichkeit wie eine Tastatur oder einen Bildschirm bieten, dem Anwender trotz-trotzdem eine Möglichkeit zur Dateneingabe sowie zur Ausgabe von Informationen bereitstellen müssen. Hier setzen die so genannten Natural Language Processing Systeme (NLP) zur Mensch-Maschine-Interaktion an.[130] Diese Systeme sollen eine Interaktion der Gesprächspartner auf Augenhöhe ermöglichen. Dabei spielt besonders die Akzeptanz solcher Systeme eine große Rolle.[131] Grundlagen sind die Verwendung von natürlicher Sprache als Bedienungssprache, die Behandlung des Systems bzw. der Maschine als gleichwertigen, sozialen Gesprächspartner und idealerweise ein domänenspezifisches Vokabular auf Seiten der Anwender und des Dialogsystems.[132]

Der für die vorliegende Arbeit relevante Teil der NLP-Systeme ist die automatische Spracherkennung (auch bekannt als Automatic Speech Recognition, ASR). Unter diesen Begriffen vereint, sind die Eigenschaften zur Erkennung verschiedener Sprecher sowie die Möglichkeit zum Erlernen neuer Wörter und damit die Erhöhung der Treffsicherheit für sprachgesteuerte Dialogsysteme im Bereich des CIoT.[133]

4.2 Chancen

Natürlichsprachliche Benutzerschnittstellen ermöglichen Anwendern zunächst einen leichten Zugang zum System. Sprache als zentrales Kommunikationsmittel der Gesellschaft ist allgegenwärtig und Regeln der Kommunikation sind schon im Kindesalter bekannt. Es ist folglich intuitiv Spracheingaben zu tätigen, um ein bestimmtes Ergebnis zu erzielen und kein technisches Vorwissen nötig.[134] Zur Verringerung der Distanz zwischen Mensch und Maschine tragen Sprachassistenten auf aktuellen Smartphones bei. Im Jahr 2016 wurden bereits auf mehr als 65% der Geräte, Suchan-

[130] Vgl. Litzel (2016).
[131] Vgl. Thar (2015), S.18.
[132] Vgl. Tahr (2015), S. 98f.
[133] Vgl. IT Wissen o.V. (2013).
[134] Vgl. Pearl (2016), S. 2f; sowie vgl. Thar (2015), S. 85.

fragen per Spracheingabe vorgenommen.[135] Die installierten Systeme sind auf modernen Smartphones Standardwerkzeuge und werden neben der Sprachsuche, auch als Diktiergeräte sowie zur Verarbeitung weiterer Sprachkommandos eingesetzt.[136] Die Entwicklung wird fortgeführt durch Sprachassistenten im Alltag, wie Smart Home Systeme, welche auch zentraler Bestandteil der Umsetzung in dieser Arbeit sind.

Darüber hinaus können durch die Schnittstelle Sprache, auch Geräte des IoT bedient werden, die keinen oder nur eine kleine visuelle Schnittstelle haben z.B. Smart Watches. Da die Bedienung ohne Einsatz der Hände möglich ist ergeben sich neue Perspektiven für den Einsatz im Kontext des autonomen Fahrens, in der Industrie zur Bedienung von Maschinen sowie bei Haushaltsgeräten.[137]

Dadurch wird deutlich, dass durch Sprachsysteme neue Geschäftsmodelle erschlossen werden können („value-add applications“).[138] Kunden nutzen diese nicht nur aus den zuvor beschriebenen praktischen Gesichtspunkten sondern auch als Unterhaltungsmedium. Dem Anwender wird eine neue Erfahrung der Interaktion geboten. Diese ist dabei nicht als Selbstzweck zu verstehen sondern als Trend in der Entwicklung in der Mensch-Computer-Interaktion.[139] Der Anwender wird dabei in den Fokus gerückt. Zukünftig sind so viele weitere Anwendungsmöglichkeiten denkbar. Wie der Hype Cycle (Abbildung 1) zeigt, befinden sich die so genannten Conversational User Interfaces, welche den Dialog zwischen Mensch und Maschine noch weiter fassen und bspw. ein vollständiges Gespräch, auch zum Transport von Stimmungen und Emotionen abbilden, noch in der Entwicklung der Erwartungsphase.

4.3 Anforderungen und Herausforderungen

Die Möglichkeiten und Chancen, die durch heutige NLP-Systeme erreichbar sind, unterliegen unterschiedlichen Anforderungen zum Erfolg. So ist die Allgemeine Nutzbarkeit der Anwendung/des Voice User Interfaces (VUI) Chance und Herausforderung zu gleich.[140] Neben funktionalen Anforderungen, die häufig als Szenarios

---

[135] Vgl. Pearl (2016), S. 3.
[136] Vgl. McGraw/et al. (2016), S. 1.
[137] Vgl. Pearl (2016), S. 3; sowie vgl. Thar (2015), S. 47.
[138] Vgl. Thar (2015), S. 47f.
[139] Vgl. Pütter (2018).
[140] Vgl. Thar (2015), S. 103f.

und Use Cases[141] dargestellt werden, gibt es weitere nicht-funktionale Anforderungen, welche zu den zentralen Eigenschaften einer automatischen Software für Spracherkennung zählen.

- Genauigkeit
- Präzision
- Latenzzeit
- Speicherverbrauch
- Speichergröße
- Sicherheit

Viele dieser Eigenschaften hängen voneinander ab, sodass der Erfüllungsgrad, je nach Anforderung der spezifischen Anwendung, eine andere Auswirkung auf die Nutzbarkeit haben kann. Unter Betrachtung des CIoT ist letztendlich die Akzeptanz beim Endanwender entscheidend. Wie im vorherigen Abschnitt beschrieben, soll ein solches Dialogsystem durch eine neue UX sowie den leichten Einstieg, ohne großes technisches Wissen erlernbar sein.

Die für den Anwender intuitiv nachzuvollziehende Eigenschaft ist die Genauigkeit. Diese zeigt, ob die natürlichsprachlichen Eingaben des Nutzers durch das ASR-System korrekt überführt werden. Die Beschreibung und Messung erfolgt häufig anhand der Word Error Rate (WER). Berechnet wird die WER durch das Verhältnis der eingegebenen Nutzerdaten, zu der vom ASR-System erkannten Transkription.[142] Das Verhältnis der fehlerhaft erkannten Eingaben wurde in den letzten Jahren, vor allem durch neuronale Netzwerke (deep neural networks, DNN), reduziert. Während lange Zeit eine WER von 15% nicht unterboten werden konnte, erreichen moderne Systeme mit DNN eine WER von unter 5%.[143] Die Präzision definiert, ob bei verschiedenen Anfragen und ähnlichem Wortstamm dasselbe Ergebnis erzielt wird. Verbessert werden kann die Präzision durch eine vorgegebene Domäne, also einem Kontextwissen des Dialogsystems und des zugehörigen Sprachmodells.[144] Die Herausforderungen der Dialogsysteme liegen, neben der stetigen Verbesserung der

[141] Vgl. Gazetic (2017), S. 24ff.
[142] Vgl. Gazetic (2017), S. 29.
[143] Vgl. Branscombe (2017).
[144] Vgl. Thar (2015), S. 50 sowie vgl. Thar (2015), S. 55f.

Spracherkennung, darin, dass Begriffe oder Fragmente der Alltagssprache unterschiedlicher Sprecher erkannt werden müssen. Außerdem soll die Eingabe auch bei lauten Umgebungsgeräuschen oder einem gestörten Übertragungsweg noch funktionieren oder zumindest nicht ohne Rückmeldung an den Nutzer abbrechen.[145]

Mit der Latenzzeit wird der Zeitraum zwischen Eingabe der Nutzerdaten und der Ausführung der zugrunde liegenden Aktion des Systems bezeichnet.[146] Für die Akzeptanz des Systems ist es besonders wichtig, dass die Systemreaktion unmittelbar ausgeführt wird. Die Herausforderung ist folglich, die Eingaben in Echtzeit zu verarbeiten, um das System als Gesprächspartner auf Augenhöhe zu etablieren und dem Anwender nicht zu suggerieren, dass verschiedene Maschinenprozesse im Hintergrund für die Interaktion durchgeführt werden müssen.[147]

Der Speicherverbrauch ist für Dialogsysteme im Umfeld des CIoT besonders zu beachten. Auch, wenn der Anwender die Auswirkungen eines Dialogsystems mit hoher Hauptspeichernutzung und CPU-Auslastung nicht zwangsläufig zu spüren bekommt, so beeinflusst diese Eigenschaft die Latenzzeit deutlich. Zur Verarbeitung der Hintergrundprozesse in Echtzeit, ist eine geringe Speicherauslastung durch das ASR-System eine Voraussetzung.[148] Dies und die Nutzung des permanenten Speichers, gilt speziell für Systeme, die vollständig offline auf einem IoT-Gerät ausgeführt werden.

Das Thema Sicherheit eines sprachgesteuerten Dialogsystems im Bereich des CIoT lässt sich sehr weit diskutieren. Neben der Transaktionssicherheit, also der störungsfreien Übertragung der Daten, soll an dieser Stelle nur der Hinweis auf den Datenschutz erfolgen. Dabei geht es um die Fragestellungen welche Eingaben des Nutzers werden gespeichert und wie werden diese Daten weiterverarbeitet.

Eine ausführlichere Darstellung des Themas und mögliche Bedrohungsszenarien sind in Kapitel 2.5 beschrieben.

---

[145] Vgl. Thar (2015), S. 54f.
[146] Vgl. Gezetic (2017), S. 28.
[147] Vgl. Thar (2015), S. 44.
[148] Vgl. McGraw/et al. (2016), S. 1.

Daneben gibt es noch weitere, nicht messbare, nicht-funktionale Anforderungen die mit der Akzeptanz des Dialogsystems einher gehen. Dazu zählt die Betrachtung der unterschiedlichen Nutzergruppen. Es gibt Erfahrenere Nutzer und Anfänger, die bisher keine bewusste Mensch-Computer-Interaktion durchgeführt haben. Es gilt also ein gemeinsames Bewusstsein für das System herzustellen. So wie in einem Gespräch, bei dem durch den Wechsel der Perspektive zur Bewertung der Hypothesen des Gesprächspartners, eine gemeinsame Basis zur Kommunikation geschaffen wird. Dies ist im Rahmen der sprachlichen Modelle eines Computers jedoch nur bedingt umsetzbar.[149]

4.4 Grundlegende Funktionsweise

In diesem Abschnitt soll die grundlegende Funktionsweise eines automatischen Spracherkennungssystems genauer betrachtet und die einzelnen Komponenten beschrieben werden. Die Funktion der ASR ist es, ein eingegebenes Sprachsignal zu extrahieren, dekodieren und mit Hilfe drei verschiedener Modelle in die textuelle Repräsentation zu überführen. Die Modelle sind das acoustic-, phonetic- sowie das language model.[150] Es werden hier bewusst die englischen Begriffe der einzelnen Modelle verwendet, da diese übergreifend in der Literatur und Praxis so genutzt werden.

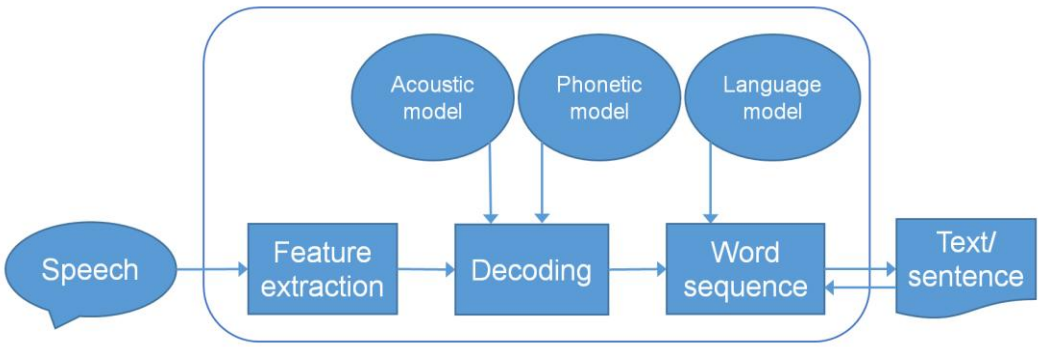

Abbildung 5: Aufbau eines ASR-Systems[151]

In Abbildung 5 ist der grundlegende Aufbau eines ASR-Systems skizziert. Zu erkennen sind die drei beschriebenen Modelle, deren Funktion nun genauer betrachtet

[149] Vgl. Thar (2015), S. 21 sowie vgl. Thar (2015), S. 108ff.
[150] Vgl. Zaman/Slany (2014), S. 373.
[151] In Anlehnung an: Militaru/Gavat (2014), S. 261.

wird. Zu Beginn startet die ASR mit dem Sprachsignal des Anwenders. Dieses Signal wird in Laute zerlegt (feature extraction).[152] Das acoustic model dient zur Zerlegung der Laute in einzelne Phoneme. Phoneme wiederum sind "die kleinste sequentielle Wortunterscheidende Lauteinheit"[153] oder semantisch betrachtet, die kleinsten Bestandteile von Silben einer Sprache.[154] Dies erfolgt mit Hilfe statistischer Verfahren. Die bekanntesten in diesem Umfeld sind die Gaussian mixture models (GMM) und das Hidden Markov Model (HMM). Die damit verbundenen statistischen Berechnungen zur Bestimmung des Phonems sollen hier nicht weiter betrachtet werden.[155] Um die Zerlegung der übertragenen Sprachsignale in einzelne Phoneme zu verbessern, werden Ausrücke in das HMM überführt. Je mehr Ausdrücke dem acoustic model zu Grunde liegen, desto treffsicherer ist dieses. In Kombination mit neuronalen Netzwerken, kann die WER weiter gesenkt werden.[156]

Das phonetic model oder phonetic dictionary hat die Aufgabe die Phoneme aus dem acoustic model, Wörtern aus einem definierten Wörterbuch zuzuordnen. Zu jedem Wort gibt es eine oder mehrere Repräsentationen in Form der zerlegten Phoneme, um verschiedene Betonungen durch unterschiedliche Sprecher zu berücksichtigen. Die angehangene tabellarische Abbildung[157] zeigt wie so ein phonetic dictionary aufgebaut sein kann. Dieses konkrete Beispiel stammt aus der Open Source Anwendung Pocketsphinx und enthält insgesamt 133.420 Wort zu Phonem-Paare für die englische Sprache. Vorteile dieser Methode sind zum einen die Praktikabilität sowie das einfache Verständnis der Modellbasis. Nachteilig ist jedoch die fehlende Genauigkeit, wenn die Domäne nicht weiter eingegrenzt werden kann. Bei kommerziellen Systemen enthält das phonetic model komplexe Algorithmen, unter Einsatz maschinellem Lernens im Rahmen der künstlichen Intelligenz, um Muster der Phoneme zu erkennen und diese Wörtern zuzuordnen.[158]

Auf Ebene mehrerer Wörter oder ganzer Sätze, wird das language model eingesetzt. Dieses definiert, welche Wörter aufeinanderfolgen können, sodass ein syntaktisch

---

[152] Vgl. Thar (2015), S. 53.
[153] Ebert/Ebert (2010), S. 179.
[154] Vgl. Ebert/Ebert (2010), S. 179f.
[155] Vgl. Militaru/Gavat (2014), S. 261f.
[156] Vgl. Lei/et al. (2013), S. 646.
[157] Vgl. Anhang A1.
[158] Vgl. CMUSphinx o. V. (o. J. a).

und semantisch korrekter Satz entsteht. Es wird auf Basis eines N-gramm-Modells bestimmt, mit welcher statistischen Wahrscheinlichkeit ein bestimmtes (Teil-)Wort auf das vorherige folgt. Dabei wird vor allem die Genauigkeit verbessert indem betonungsverwandte, aber semantisch verschiedene Folgewörter eines Satzes, ausgeschlossen werden können.[159] Auf Satzebene unterstützen N-Best-Listen die Identifizierung der vom Benutzer getätigten Eingaben genauer. Diese Listen basieren auf Text- oder Sprachdatenbanken des Anwendungssystems und verknüpfen die Wahrscheinlichkeiten der Wort- und Satz-Hypothesen mit bereits bekannten Datensätzen.[160]

4.5 Vergleich der Technologien

4.5.1 Online

4.5.1.1 Funktion und Beispiele

Im Bereich der Online-Systeme zur Verarbeitung natürlichsprachlicher Eingaben sind kommerzielle Systeme führend. Prägendste Eigenschaft dieser Systeme ist die dezentrale Verarbeitung der Nutzerdaten auf Servern.

Kommerzielle Systeme werden häufig als Cloud-Dienste angeboten, lassen sich entsprechend leicht skalieren und die Kosten für die Verwendung realistisch abschätzen. Beispiele sind Google Cloud Speech, Microsoft.Speech sowie Amazon Transcribe. Gemeinsamkeit dieser ASR-Systeme ist die Bereitstellung von APIs für verschiedene Programmiersprachen und Betriebssysteme zur Anbindung an beliebige Anwendungen.

Das Training der Modelle liegt dabei größtenteils auf Seiten der Anbieter des Services. Die stetige Verbesserung der im Hintergrund genutzten Modelle erfolgt mittels Funktionen maschinellen Lernens durch neuronale Netze.[161]

---

[159] Vgl. Thar (2015), S. 53.
[160] Vgl. CMUSphinx o. V. (o. J. a) sowie vgl. Thar (2015), S. 53.
[161] Vgl. Retresco o. V. (2017).

4.5.1.2 Vor- und Nachteile

Die Vor- und Nachteile eines ASR-Systems, Online und Offline, lassen sich anhand der beschriebenen Anforderungen eines solchen Systems darstellen. Als einer der größten Vorteile eines Online-Systems gilt die Nutzung und Verarbeitung der eingegebenen Sprachdaten in Echtzeit. Die eingegebenen Daten können entweder manuell oder durch ein automatische Algorithmen neuronaler Netze analysiert und zur Verbesserung der Genauigkeit des Systems beigetragen werden.[162]

Darüber hinaus ist die Anforderung an Speicherressourcen auf dem Gerät geringer. Die Auswertung des Sprachsignals erfolgt auf Servern in Rechenzentren, sodass weder eine besonders starke CPU, noch ein großer Arbeitsspeicher zur Ausführung auf dem Gerät benötigt wird. Des Weiteren ist die Nutzung des permanenten Speichers geringer, da keines der im vorherigen Kapitel beschriebenen Modelle und Wörterbücher auf dem Gerät installiert sein muss.

Damit die dezentrale Verarbeitung der Nutzerdaten erfolgen kann, ist eine Netzwerkverbindung Grundvoraussetzung. Je nach Verfügbarkeit und Netzausbau kann die Verbindung zum Server also gestört oder gar nicht vorhanden sein. Bei einer gestörten Verbindung erhöht sich die Latenzzeit deutlich und die Verarbeitung in Echtzeit ist nicht weiter gewährleistet. Bei einer vollständigen Störung oder nicht vorhandenen Netzwerkverbindung sind alle Anwendungsfälle nicht ausführbar.[163]

Abschließend ist das Thema Sicherheit und Datenschutz bei vollständig Online operierenden Systemen ein sehr großes. Der Nutzer hat wenig Einfluss darauf, welche seiner Daten tatsächlich an das Rechenzentrum übertragen werden und wann sein ASR-System aktiv ist. Die Risiken und Bedrohungsszenarien die hierdurch entstehen können sind in Kapitel 2.5 bereits erläutert.

---

[162] Vgl. McGraw/et al. (2016), S. 1.
[163] Vgl. McGraw/et al. (2016), S. 1.

4.5.2 Offline

4.5.2.1 Funktion und Beispiele

Wie dargelegt, unterscheidet sich die in Abbildung 5 skizzierte Funktionsweise eines Offline-Systems, nicht von der eines Online-Systems. Die Besonderheit dieser Systeme ist, dass sie fest auf einem Gerät installiert sind. Die Anwendungen werden als gesamtes System mit allen Modellen und Wörterbüchern ausgeliefert. Dementsprechend ist das permanente Training der acoustic und language models in Echtzeit mit den Nutzereingaben nicht möglich. Die Bedeutung des Trainings dieser Modelle vor der Auslieferung, hat bei Offline ASR-Systeme also eine große Bedeutung.

Das kommerzielle Offline-System zur automatischen Spracherkennung von Google nutzt für das Training des acoustic models mehr als 3 Millionen Ausdrücke aus bereits durchgeführten Suchanfragen. Dabei werden zusätzlich Audiodateien aus YouTube-Videos extrahiert und im Hintergrund abgespielt, um 20 verschiedene (verzerrte) Aufnahmen des selben Ausdrucks verarbeiten zu können und eine Robustheit gegen Hintergrundgeräusche zu erzielen.[164] Um den Fußabdruck der Anwendung, also den Verbrauch des Speicherplatzes und der Rechenleistung gering zu halten, werden die einzelnen Modelle komprimiert, damit auch Geräte mit geringem permanenten Speicher diese einsetzen können.[165]

Open-Source Systeme wie CMU Sphinx profitieren ebenfalls von umfangreichen Audio-Testdaten. Modelle für verschiedene Sprachen werden als Download bereitgestellt. Allerdings unterschieden sie sich in dem Punkt von kommerziellen Systemen, dass zur Erreichung einer hohen Genauigkeit ein zusätzliches Training durch den Entwickler notwendig ist. Allerdings kann die Beschränkung auf eine bestimmte Domäne und damit einem selbst definierten Wörterbuch, die Genauigkeit erhöht werden.[166] Weitere bekannte Offline-Systeme sind Kaldi und Snowboy.

---

[164]Vgl. McGraw/et al. (2016), S. 2.
[165] Vgl. McGraw/et al. (2016), S. 3.
[166] Vgl. CMUSphinx o. V. (o. J. b).

4.5.2.2 Vor- und Nachteile

Die Vor- und Nachteile von Offline und Online Systemen zur automatischen Spracherkennung sind nicht komplett komplementär. Jedoch wird bei der Betrachtung der nicht funktionalen Anforderungen und deren aktuelle Ausprägung bei Offline ASR-Systemen deutlich, dass die Systeme unterschiedliche Anwendungsfälle adressieren.

Das größte Hemmnis der Offline-Anwendungen auf einem IoT-Gerät ist aktuell die Genauigkeit. Die Entwicklung der WER in den vergangenen Jahren hat gezeigt, dass sowohl Online- als auch Offline-Systeme deutliche Fortschritte erzielt haben. Die erzielten Benchmarks und dafür eingesetzten Methoden differieren zwischen den Anbietern.[167] Versucht man die Daten eines Herstellers für Offline- und Online ASR-Systeme zur ungefähr gleichen Zeit in Beziehung zu setzen, zeigt die WER der Google Systeme einen Unterschied von ungefähr 5% (Offline: 13,5% WER; Online: 8,5% WER).[168] Andere Untersuchungen zeigen zudem, dass Open Source Systeme, in diesem Fall CMU Sphinx, eine WER von 37% besitzen. Das allerdings basierend auf nicht domänenspezifisch eingegrenzten models und dictionaries.[169]

Für ein Offline ASR-System ist jedoch keine (dauerhafte) Netzwerkverbindung erforderlich, um Spracheingaben zu tätigen. Der Anwender ist somit unabhängiger von der benötigten Infrastruktur. Da die ad-hoc Auswertung von Nutzereingaben zur automatischen Verbesserung der Genauigkeit nicht möglich ist, müssen beim Anwendugs- und Produktionskonzept bereits Faktoren, wie automatische Updates z.B. über einen zentralen App Store oder over-the-air berücksichtigt werden. Daher ist die Analyse von Nutzerdaten bei der Erstellung der models und dictionaries extrem wichtig.

Des Weiteren werden mehr Ressourcen der CPU und des Arbeitsspeichers benötigt, da die Dekodierung und Verarbeitung des eingegebenen Signals auf dem Gerät erfolgt. Gleiches gilt für den permanenten Speicher, der neben der Anwendung die unterschiedlichen models und dictionaries persistieren muss. Durch den direkten

---

[167] Vgl. Branscombe (2017).
[168] Vgl. Lei/et al. (2013), S. 664 sowie vgl. Protalinski (2017).
[169] Vgl. Këpuska/Bohuta (2017), S. 23.

Zugriff und die nicht obligatorische Netzwerkverbindung kann aber eine geringere Latenzzeit und damit ein schnellerer Zugriff erzielt werden.[170]

4.6 Dialogverarbeitung

Zentrales Element der Verarbeitung von Nutzereingaben, nachdem das Sprachsignal durch die zuvor beschriebenen Mechanismen in seine textuelle Repräsentation überführt wurde, ist die Dialogverarbeitung (auch: Dialogmanager). Die Aufgaben des Dialogmanagers sind das Halten des Zustands der Kommunikation des Anwenders mit dem Dialogsystem, die Steuerung des Dialogflusses sowie die Zuordnung der eingegebenen Daten zu bestimmten Events. Diese lösen dann weitere Funktionen des Anwendungssystems (z.B. Datenbankzugriffe) aus.[171] Zu den weiteren Aufgaben und zur Erhaltung des Dialogflusses, zählen das Aktivieren von Events zum Rückmelde- und Bestätigungsverhalten des Systems gegenüber dem Anwender. Darüber hinaus die Bereitstellung einer Hilfe-Funktion und das behandeln von Fehlersituationen, sodass die Interaktion des Benutzers mit dem Dialogsystem nie (ohne Rückmeldung) unterbrochen wird.[172]

Der grundsätzliche Aufbau der Dialogverarbeitung ist angelegt als die Zustandsbeschreibung eines Automaten. Dabei wird der Umgang mit Eingaben des Nutzers vorab im System hinterlegt. Es wird sichergestellt, dass auf jede Eingabe auch eine entsprechende Rückmeldung erfolgt. Diese kann wie beschrieben im Erfolgsfall eine Aktion im Anwendungssystem auslösen, Nachfragen zur genaueren Spezifizierung der Nutzereingaben durch den Dialogmanger zur Folge haben oder im Fehlerfall eine Hilfe-/Fehlermeldung ausgeben.[173]

Bekannte Systeme zur Dialogverarbeitung in der Praxis, speziell im Bereich des CIoT, sind das Alexa Skills Kit, Actions on Google sowie der ebenfalls zu Google gehörende Dialogmanager Dialogflow (früher: API.ai). Alle Systeme werden über eine definierte Aktion aufgerufen. Das kann durch Betätigen eines Schalters, durch Bewegung oder wie bei Smart Home Systemen häufig, durch ein Schlüsselwort (Wakeword: "Alexa", "OK, Google") ausgelöst werden (Invocation). Zur anschlie-

[170] Vgl. Lei/et al. (2013), S. 663f.
[171] Vgl. Schnelle-Walka/et al. (2016), S. 1
[172] Vgl. Thar (2015), S. 56.
[173] Vgl. Thar (2015), S. 56 sowie vgl. Schnelle-Walka/et al. (2016), S. 2f.

ßenden Verarbeitung der natürlichsprachlichen Eingaben werden so genannte "Intens" und "Slots" eingesetzt. Die Modellierung der Intents, welche die durch den Anwender gewünschte Funktion auslösen und Slots, welche Wörter einer spezifischen Domäne enthalten und Intents zugeordnet werden, sind für die Nutzbarkeit der Anwendung entscheidend. Für die Weiterverarbeitung (Fulfillment) des ausgelösten Intents bieten die Dialogmanager-Systeme APIs für verschiedene Programmiersprachen an.[174]

[174] Vgl. Amazon o. V. (o. J. b); Vgl. Google o. V. (2018) sowie vgl. Dialogflow o. V. (2018a).

5 Konzeption des Smart Home Systems

5.1 Umfang

Aufbauend auf den vorangegangenen Ausführungen soll in diesem und dem Folgenden Kapitel die Konzeption und Entwicklung eines Smart Home Prototyps stattfinden. Im Umfang dieser Arbeit ist es nicht Ziel, alle vorgestellten Treiber und Technologien umzusetzen. Allerdings basieren die demonstrierten Funktionen und Anwendungsfälle auf den dargestellten theoretischen Grundlagen.

Das Entwicklungsvorgehen entspricht dem Prototyping in der Form, dass ein gesamter technischer Durchstich, aller relevanten Technologien sowie deren Integration erfolgt. Es handelt sich also um einen vertikalen Prototyp, der den Fokus mehr auf der technischen Realisierung hat, als auf fachlich vollständig ausgearbeiteten Anwendungsfällen.[175] Gleichwohl gilt, dass grundlegende Anforderungen spezifiziert werden und der Qualität, hinsichtlich der Umsetzung und Erweiterbarkeit des Systems, eine wichtige Bedeutung zukommt.[176] Unter Betrachtung der eingesetzten Systembestandteile, die nachfolgend detailliert beschrieben werden, lässt sich das System in der Taxonomie nach Floyd, unter dem "experimentellen Prototyping" einordnen.[177]

Grundlegende Eigenschaften der Anwendung werden nachfolgend spezifiziert. Ein strenges Vorgehen der Anforderungsermittlung nach Vorgaben des Requirement Engineering stehen nicht im Fokus und sind zur Erreichung der Zielsetzung auch kein relevanter Bestandteil. Nicht-funktionale Anforderungen, durch Vorgaben der Zielsetzung dieser Arbeit, werden in den einzelnen Abschnitten des Kapitels 5.3 genauer beleuchtet.

Ebenfalls nicht im zentraler Bestandteil der Konzeption, ist das Design einer visuellen Schnittstelle (Graphical User Interface, GUI). Die Benutzung des Systems soll ohne grafische Oberfläche möglich sein. Die umgesetzten GUIs dienen lediglich der einfachen Testbarkeit sowie zu Demonstrationszwecken.

---

[175] Vgl. Kleuker (2018), S. 29ff.
[176] Vgl. Ludewig/Lichter (2013), S. 170.
[177] Vgl. Ludewig/Lichter (2013), S. 168f.

5.2 Funktionen und Anwendungsfälle

Die tabellarische Abbildung zeigt die Funktionen und Anwendungsfälle, die in der Konzeption und Entwicklung des Smart Home Prototyps genauer betrachtet werden.

Funktion/ Anwendungsfall	**Beschreibung**
Wakeword	Die Aktivierung des Smart Home Systems soll über das Schlüsselwort "My Assistant" erfolgen. Eine weitere Möglichkeit der Aktivierung ist nicht vorgesehen.
Spracherkennung	Die Bedienung des Smart Home Systems soll mittels natürlichsprachlicher Eingaben erfolgen. Dazu ist die Spracherkennung durch ein ASR-System umzusetzen.
Dialogverarbeitung	Intelligente Verarbeitung der Spracheingaben durch automatische Erkennung des Anwendungsfalls und seiner Spezifika sowie Rückmeldung zur Weiterverarbeitung der Nutzerabsicht in der Smart Home Anwendung.
Sensordaten	Auslesen der Sensordaten: Ermittlung der aktuellen Temperatur sowie des Luftdrucks mittels Sensor. Publishing der Sensordaten: Die erhobenen Daten sollen über den Pub/Sub-Mechanismus an den Cloudspeicher übergeben werden.
Begleitanwendung	Empfang von Meldungen: Bei einer plötzlichen Temperaturerhöhung seitens des Temperatursensors, soll eine automatische Benachrichtigung an die Begleitanwendung auf dem Smartphone des Anwenders gesendet werden. Einstellung der Temperatur: Der Anwender soll die Einstellung der Raumtemperatur (virtuell) über die Begleitanwendung vornehmen können. Die Folge ist eine automatische Verarbeitung der Information durch das Smart Home System.

Abbildung 6: Übersicht der Funktionen und Anwendungsfälle

Es wird zwischen Funktionen und Anwendungsfällen unterschieden. Unter Funktionen werden im Folgenden alle Features des Systems gefasst, die Basisaufgaben einer sprachgesteuerten Smart Home Anwendung abbilden (Aktivierung durch das Schlüsselwort, automatische Spracherkennung, intelligente Dialogverarbeitung). Anwendungsfälle hingegen sind konkrete Beispiele der Nutzung, die von System zu System eine unterschiedliche Ausprägung haben können (Auslesen und Übertragen der Sensordaten, Meldungen an die Begleitanwendung sowie Steuerung des Systems per Begleitanwendung).

Die zentralen nicht-funktionalen Anforderungen der Anwendung, wie Geschwindigkeit und Korrektheit der Verarbeitung, Fehlertoleranz, Flexibilität und Bedienbarkeit, werden in den Abschnitten zu Grundlagen und Technologien aufgegriffen.

Neben den in Abbildung 6 aufgelisteten Funktionen und Anwendungsfällen, werden Dummy-Implementierungen für zusätzliche Funktionen eingesetzt. Diese sind notwendig, um Grundlagen für Dialogsituationen bereitzustellen und eine Integration mit den umgesetzten Anwendungsfällen zu ermöglichen z.B. Festlegung der Raumtemperatur mittels Spracheingabe im Smart Home System und Anpassung durch die Begleitanwendung.

5.3 Grundlagen und Technologien

5.3.1 Hardware

Plattform für die Ausführung des Smart Home Systems ist der Raspberry Pi 3 Model B. Dieser Einplatinencomputer besitzt einen 64-bit Quadcore Prozessor mit einer Taktrate von 1,4GHz, 1GB Hauptarbeitsspeicher und unterstützt alle gängigen Kommunikationsschnittstellen, darunter die für das IoT wichtigen WLAN-Spezifikationen und BLE. Darüber hinaus verfügt der Raspberry über 4 USB 2.0 Anschlüsse, einen HDMI-Anschluss und 40 von außen adressierbare General Purpose Input/Output Kontakte (GPIO-Pins).[178] Zur permanenten Speicherung des Betriebssystems sowie weiterer Anwendungen und Anwendungsdaten wird die MicroSD-Schnittstelle genutzt.

[178] Vgl. Raspberry Pi Foundation o. V. (o. J.), S. 2f.

Weitere Hardware-Bestandteile des Smart Home Systems sind ein USB-Mikrofon zur Aufnahme der Spracheingaben des Anwenders und ein Lautsprecher, welcher über USB mit Strom versorgt wird und mit dem Audio-Ausgang des Raspberry, durch einen Klinkenstecker verbunden ist.

Basis für den Anwendungsfall Sensordaten, ist das Mikrocontroller Modul BMP280 von Bosch. Der Sensor misst mit einer Genauigkeit von +/- 1hPa sowie 1°C[179], was für die Umsetzung der Anwendungsfälle des Prototyps ausreichend ist.

Für die Umsetzung der Anwendung nicht zwangsläufig benötigt, aber zur Vereinfachung der Entwicklung und Durchführung der Entwicklertests wichtig, ist das offizielle Raspberry Pi 7 Zoll Touchscreen Display aufzuführen.

Zusammengefasst bietet der Raspberry PI für die gestellten Anforderungen an das Smart Home System eine sehr gute Plattform. Die Rechenleistung, Flexibilität des Speichers sowie die verschiedenen Anschluss- und Kommunikationsschnittstellen bekräftigen diese Wahl.

5.3.2 Betriebssystem - Android Things

Als Basisbetriebssystem für die Laufzeit der Implementierung wird Android Things verwendet. Android Things ist ein, nach aktuellem Stand, noch in Entwicklung befindliches, auf Android basierendes IoT-Betriebssystem. Die genutzte Version der Entwicklervorschau ist Android Things Developer Preview 8. Diese Version ist die aktuellste Version und gilt als Release Kandidat für die erste marktreife Fassung.[180]

Viele der grundlegenden Funktionalitäten des Mutter-Betriebssystems Android, deckt auch das speziell für das Internet der Dinge entwickelte Android Things ab. Dies ist dargestellt in Abbildung 7, durch die beiden Blöcke "Java API Framework" und "Google Services". Eine für die Umsetzung relevante, fehlende Funktion, ist das auf Smartphones eingesetzte Offline ASR-System von Google. Des Weiteren sind nicht alle Funktionen der Google Play Services nutzbar z.B. Google Drive (Cloudspeicher für Anwender) und Google Pay (Onlinebezahlsystem).[181] Dafür bietet das Betriebssystem verschiedene Treiber zur Unterstützung unterschiedlicher

[179] Vgl. Adafruit o. V. (2015).
[180] Vgl. Smith (2018).
[181] Vgl. Android Developers o. V. (2018a).

Module des IoT an. In der Abbildung repräsentiert durch die "Things Support Library". Dazu zählen Treiber für Sensoren (z.B. Temperatur und Luftdruck) sowie für Übertragungsstandards (z.B. Bluetooth Low Energy, BLE). Die Anwendungen ("Apps") können diese Funktionen des Android SDKs beliebig einsetzen.[182]

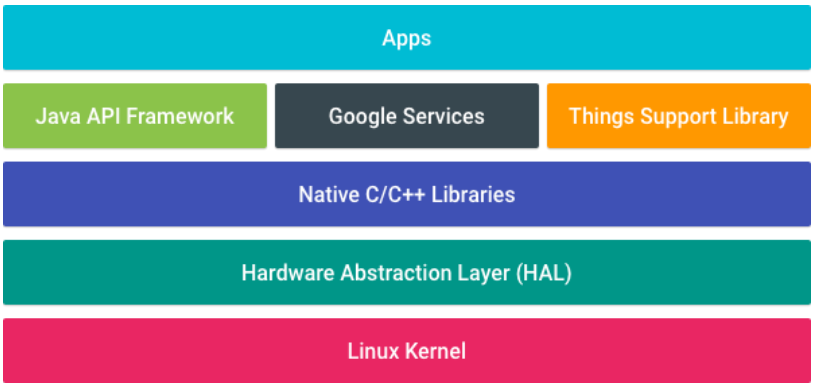

Abbildung 7: Android Things Plattform Architektur[183]

Besonderheit des Betriebssystems ist zudem, dass Berechtigungen z.B. Zugriff auf den permanenten Speicher, Netzwerkverbindungen oder externe Geräte, wie ein Mikrofon, bereits beim Start der Anwendung automatisch vergeben werden. Hierfür müssen diese in die Manifest-Datei der Anwendung eingetragen werden. Im Normalfall ist die explizite Bestätigung durch den Anwender beim Start der App erforderlich. Da Geräte des IoT jedoch nicht zwangsläufig eine grafische Schnittstelle besitzen, wird die Berechtigungsvergabe über diese Konfiguration durchgeführt. Als Entwicklungsumgebung wird Android Studio eingesetzt. Die auf IntelliJ basierende, integrierte Entwicklungsumgebung (Integrated Development Enviromnent, IDE) unterstützt die Abhängigkeitsverwaltung mit Gradle, die Versionsverwaltung (z.B. Git) sowie den Test und die Auslieferung der Anwendung.[184]

Neben dem in dieser Arbeit verwendeten Raspberry Pi 3 Model B, unterstützt Android Things weitere Einplatinencomputer z.B. den NXP Pico i.MX7D und den NXP Argon i.MX6UL.[185] Das Betriebssystem sowie die Anwendungen werden auf dem externen Speicher (MicroSD-Card) des Raspberry installiert.

---

[182] Vgl. Android Developers o. V. (2018b).
[183] Android Developers o. V. (2018b).
[184] Vgl. Android Developers o. V. (o. J.).
[185] Vgl. Android Developers o. V. (2018c).

Die Entscheidung für Android Things ist auf der einen Seite durch die Motivation begründet, neue Technologien zu entdecken und Know-How aufzubauen. Auf der anderen Seite aber eine rationale Entscheidung, da auf einer etablierten Plattform aufgesetzt wird. Entwicklungsmodelle, SDKs, spezielle Treiber-Bibliotheken und nicht zuletzt, eine große Community der Entwickler, sind für diese Entscheidung anzuführen.

5.3.3 Dialogsystem

5.3.3.1 Wakeword - Pocketsphinx

Das Werkzeug zur Unterstützung der Funktion Wakeword, ist CMU Pocketsphinx. Pocketsphinx ist ein Teil des CMU Sphinx-Toolkits, welches neben dieser leichtgewichtigen Umsetzung (speziell für mobile Anwendungen), noch weitere Funktionen zur automatischen Spracherkennung und dem Training der Modelle bereitstellt.[186]

Es handelt sich um ein offline ASR-System und liefert Ressourcen aller benötigten Modelle und Wörterbücher zur Ausführung auf dem Gerät. In diesem Fall ist die von CMU Pocketsphinx genutzte Version eine dedizierte Fassung für Android. Die Besonderheit dieser Version ist, die Bereitstellung der benötigten Funktionen über eine Android Archive Library (*.aar). Diese Bibliothek ist zu vergleichen mit einem Java Archive (*.jar), sie existiert als selbstständige Android-Anwendungen und kann neben einer eigenen Build-Konfiguration auch eigene Ressourcen enthalten.[187]

Die Modelle und Wörterbücher eines ASR-Systems, welche in Kapitel 4.4 vorgestellt wurden, stehen in grundlegender Funktion beim Hersteller zum Download bereit und können direkt in die Anwendung integriert werden. Das zu nutzende Sprachmodell ist ausgelegt für die englische Sprache, mit einem großen phonetischen Wörterbuch von 133.420 unterschiedlichen Wort-zu-Phonem Variationen.[188]

Eine weitere Eigenschaft von Pocketsphinx ist, dass es eine always-on Funktionalität bietet. Im Kontext der Spracherkennung wird damit das dauerhafte Lauschen auf ein Wakeword/Schlüsselwort beschrieben. Die Aktivierung des Smart Home Prototyps soll über das Wakeword "My Assistant" erfolgen. Entwicklertests der Beispielan-

[186] Vgl. CMUSphinx o. V. (o. J. c).
[187] Vgl. CMUSphinx o. V. (o. J. d).
[188] Vgl. Anhang A1.

wendung[189] von Pocketsphinx haben gezeigt, dass die Erkennung des Wakewords präzise funktioniert, wenn dieses eine Länge von zwei bis drei Wörtern besitzt, die einen phonetisch unterschiedlichen Stamm haben.

Die Entscheidung für Pocketsphinx zur Erkennung des Wakewords ergibt sich somit aus vielerlei Gesichtspunkten. Zentral, sind die speziell für Android-Systeme bereitgestellte Version und die Möglichkeit zur Vergabe eines benutzerdefinierten Wakewords. Die Untersuchung der Beispielanwendung hat gezeigt, dass das System bei akzeptabler Speicher- und CPU-Auslastung betrieben werden kann und keine hohen Latzenzzeiten entstehen. Als abschließender Aspekt ist das Thema Datenschutz anzuführen. Durch die vollständige Offline-Funktionalität findet die gesamte Verarbeitung auf dem System statt. Damit bleibt die Kontrolle der erhobenen Daten, durch die always-on Funktion des Systems, vollständig auf Seiten des Prototyps. Die Berücksichtigung der Prinzipien "Privacy by Design" und "Privacy by Default" aus Art. 25 der neuen EU-DSGVO, werden somit in der Konzeptionsphase der Anwendung durchgesetzt.

Die Umsetzung der gesamten Spracherkennung mit Pocketsphinx ist allerdings nicht praktikabel, da sowohl Untersuchungen mit der Beispielanwendung als auch andere Untersuchungen gezeigt haben, dass die WER bei Offline-Systemen allgemein (siehe Kapitel 4.5.2.2) und auch bei Pocketsphinx im speziellen, zu hoch ist.[190]

5.3.3.2 Spracherkennung - Cloud Speech-To-Text

Cloud Speech-To-Text ist eine Funktion aus dem Cloud-Service Cloud Platform von Google. Zentrale Eigenschaften sind die Erkennung von gesprochener Sprache in Echtzeit oder auf Basis von Audiodateien (ASR). Die automatische Spracherkennung wird durch neuronale Netzwerke stetig weiterentwickelt, ohne dass der Anwender eigene Algorithmen oder Modelle entwickeln muss. Einflussmöglichkeit auf die im Hintergrund genutzten Modelle, besteht durch Bereitstellung von Worthinweisen eines bestimmten Kontexts, z.B. Navigation, oder durch Erweiterung dieser, mit be-

[189] Vgl. CMUSphinx o. V. (o. J. e).
[190] Vgl. McDonley (2016).

nutzerdefinierten Wörtern.[191] Die integrierte API wird für verschiedene Programmiersprachen bereitgestellt, dazu zählen Java, Python, Node.js, C++ und weitere.

Da es sich bei Cloud Speech-To-Text um ein kommerzielles System handelt, ist eine Registrierung bei der Google Cloud Platform erforderlich. Wie bei vielen Cloud Services üblich, wird die Nutzung über die Dauer der transkribierten Eingaben bepreist. Für Entwickler gibt es ein freies Volumen, um Funktionen der API, zur Prüfung der Anforderungen an die eigene Anwendung im Rahmen des Prototypings, auszuprobieren.[192]

Der Einsatz der Google Cloud eignet sich für den vorliegenden Prototyp besonders gut, aufgrund des für die Android-Entwicklung bereitgestellten SDKs. Da weitere Cloud Services für die Erstellung des Prototyps eingesetzt werden, ist die Integration dieser Services untereinander ebenfalls ein wichtiger Entscheidungspunkt (vgl. hierzu auch Abschnitt 5.3.5). Speech-To-Text ist die Basis diverser Google Services sowie weiterer Dritthersteller. Die breite produktive Nutzung verdeutlicht die Marktreife des Produkts und bietet eine zukunftsfähige Grundlage für die Weiterentwicklung des Smart Home Prototyps in mehreren Sprachen. Zudem ist keinerlei weiteres Training der Modelle oder Erweiterung der Wörterbücher notwendig. Die gesamtheitliche Nutzung, ist vor allem daher nicht möglich, da es keine Möglichkeit zur Vergabe eines benutzerdefinierten Wakewords gibt.

5.3.3.3 Dialogverarbeitung - Dialogflow

Das Kernelement eines Dialogsystems ist die Dialogverarbeitung, auch Dialogmanager genannt. Im Umfang des Prototyps wäre eine manuelle Verarbeitung der Spracheingaben auf Basis bestimmter Wörter zu einzelnen Domänen denkbar. Allerdings zeigt sich schnell, dass nicht alle Spracheingaben des Benutzers, einer bestimmten Absicht zugeordnet werden können. Die intelligente Verarbeitung der Nutzerdaten und das Erzeugen eines Objekts zur anschließenden Weiterverarbeitung wird mit Dialogflow umgesetzt. Dialogflow ist ein auf der Google Cloud Platform ausgeführtes Werkzeug zum Modellieren und Verarbeiten von Dialogsituationen zwischen Mensch und Maschine.

---

[191] Vgl. Google Cloud o. V. (o. J. a).
[192] Vgl. Google Cloud o. V. (o. J. a).

Als Fundament der UX ist die Orientierung an der Unterhaltung zwischen zwei Personen grundlegender Bestandteil und bildet damit den in Kapitel 4.6 aufgestellten Grundsatz, zur Kommunikation auf Augenhöhe zwischen Mensch und Maschine, ab. Die genaue Funktionsweise soll am Beispiel einer Unterhaltung über das Wetter verdeutlicht werden.[193]

Die Begrüßung zwischen zwei Personen erfolgt durch ein Grußwort "Hallo Julius", "Hi Theresa". Auf Seiten des Dialogsystems wird dies durch das Wakeword "My Assistant" umgesetzt. Der Beginn der Unterhaltung wird als Invocation bezeichnet.

Im Anschluss stellt Person A ("Julius"), Person B ("Theresa") ein Frage: "Wie wird das *Wetter morgen* in *Münster* werden?". Theresa versteht die Semantik dieser Frage und sucht Informationen zu den morgigen Wetterbedingungen, über eine Quelle ihrer Wahl heraus. Die Verarbeitung dieser beiden Schritte im Dialogmanager ist deutlich komplexer. Zunächst muss Dialogflow die eingegebene Frage (Intent) parsen. Im vorhinein müssen durch den Entwickler so genannte Entities angelegt und mit unterschiedlichen Begriffen und Phrasen der Domäne trainiert werden. Dadurch erkennt Dialogflow automatisch, welche Begriffe semantisch zu welcher Entity gehören. Beim parsen würde "*Münster*" bspw. der Entity "Stadt" und "*morgen*" der Entity "Datum" zugeordnet werden. Anhand dieser Informationen kann die Weiterverarbeitung über einen Webhook eines Web-Services, oder durch lokale Verarbeitung mittels API-Aufruf durchgeführt werden (Fulfillment).

Das Resultat der Suche teilt Theresa Julius persönlich mit. Im Falle des Dialogmanagers wird die Antwort entweder durch Dialogflow selbst oder durch eine zugrunde liegende Anwendung per Text-To-Speech ausgegeben (Response). Im vorliegenden Szenario werden die Schritte Fulfillment und Response durch den Smart Home Prototyp umgesetzt.

Die Vorteile von Dialogflow sind zum einen die einfache Integration mit anderen Funktionen der eingesetzten Google Cloud Platform. Zum anderen wird ein Android SDK zur Anbindung an das Smart Home System angeboten. Für die prototypische Entwicklung ist es zudem wichtig, auf bereis existierende Trainingsmechanismen des maschinellen Lernens zurückgreifen zu können, um schnell und einfach Ergebnisse

[193] Vgl. Dialogflow o. V. (2018a).

zu erzielen. Abschließend ist die Erweiterbarkeit ein großer Vorteil. Dialogflow bietet Schnittstellen zu verschiedenen sprach- oder textbasierten Kommunikationsanwendungen an z.B. Skype, Facebook Messenger oder dem Google Assistant.[194] Zudem können vorgefertigte Manager (Prebuilt Agents) z.B. zur Abfrage des Wetters oder zur Währungsumrechnung eingesetzt werden. Dies erleichtert die Entwicklung zukünftiger Anforderungen und bietet die Möglichkeit Anwendungen von Drittherstellern in das System mit einzubeziehen.

5.3.3.4 Dialogdesign

Eine detaillierte Betrachtung der Prinzipien für das Design eines VUI liegen nicht im Fokus der technischen Erstellung des Smart Home Prototyps und würden über den Umfang der Arbeit hinausgehen. Trotzdem soll eine kurze Einführung in Fragestellungen und Methodiken, zur Konzeption einer natürlichsprachlichen Schnittstelle, vorgestellt werden.

Zu den grundlegenden Fragestellungen zählt, für welche Nutzergruppe soll das System entwickelt werden und wie häufig setzen sie dieses ein? Da der Prototyp einen experimentellen und eher technischen Hintergrund besitzt, lässt sich die Nutzungsgruppe auf Entwickler und technikaffine Anwender eingrenzen. Falsch interpretierte oder fehlerhafte Eingaben, die zum Absturz der Anwendung oder einem Deadlock der Dialogsituation führen, sollen möglichst ausgeschlossen werden, sind aber kein Merkmal des Scheiterns hinsichtlich der Umsetzung. Damit wird auch die Frage nach der Qualität der ausgegebenen Antworten, im Bezug auf die Professionalität des Geschäfts, beantwortet. Da zunächst kein Endanwenderkontakt vorgesehen ist, gilt es die Variabilität und Ausformulierung der Antworten zu vernachlässigen. Bei einem Dialogsystem zur Buchung eines Hotelzimmers würde diese Frage selbstverständlich anders beantwortet werden. Der Kunde soll sich, trotz automatisierter Steuerung, in seinen Anforderungen und Wünschen verstanden fühlen und eine entsprechende Rückmeldung erhalten.

Andere Fragen zielen auf die Umsetzung des Dialogmanagers ab. Sind genügend Beispielphrasen je Intent vorhanden? Unterscheiden sich die Phrasen voneinander? Werden nach dem Fulfillment Informationen des Kontexts gehalten, sodass An-

---

[194] Vgl. Dialogflow o. V. (2018b).

schlussfragen gestellt werden können, ohne dass eine erneute Invocation erzeugt werden muss?[195]

Um diese Fragen zu beantworten werden Dialogbäume eingesetzt. Nachfolgend dargestellt ist der Dialogbaum für die Einstellung der Heizung zur Regulierung der Raumtemperatur, welche im Rahmen des Prototyps umgesetzt wird.[196]

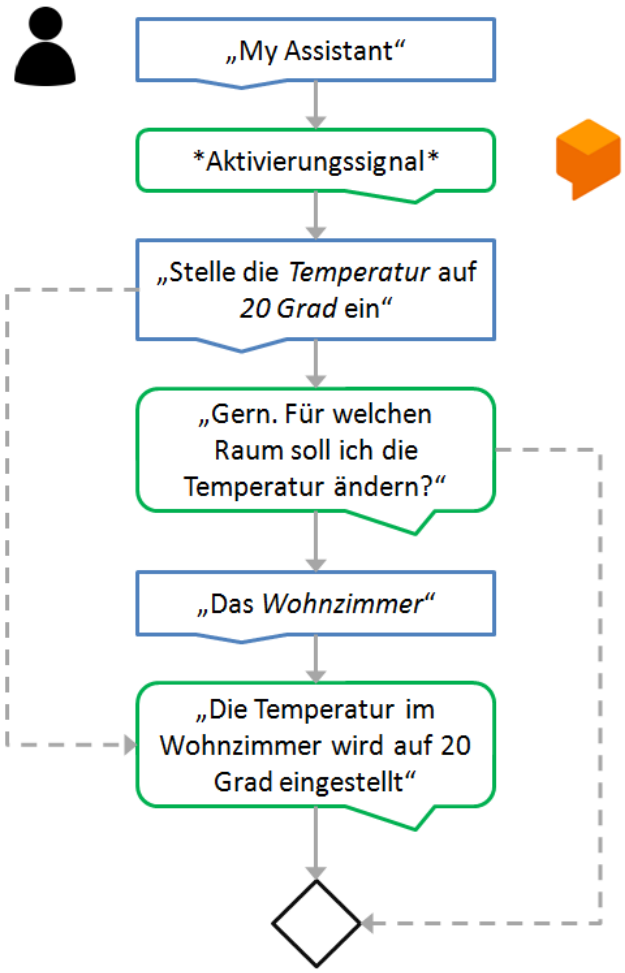

Abbildung 8: Dialogbaum Steuerung der Raumtemperatur

Der skizzierte Dialogbaum veranschaulicht einen möglichen Dialog zwischen Anwender und System. In blau mit der stilisierten Figur dargestellt, sind die Eingaben des Nutzers. In grün mit dem Dialogflow-Logo, die des Dialogmanagers. Erkennbar sind die unterschiedlichen Elemente des Dialogflow-Sprachagenten. "My Assistant" ist die Invocation, "Stelle die Temperatur auf 20 Grad ein" ist der Intent mit den bei-

[195] Vgl. Dialogflow o. V. (2018c).
[196] Vgl Thar (2015), S. 74 sowie vgl. Dialogflow o. V. (2018d).

den kursiv dargestellten Entities "Temperatur" und "20 Grad". Bei nur einem Raum wird direkt die Response aufgerufen (linke gestrichelte Linie), bei mehreren Räumen ist der Dialog noch nicht beendet und der Dialogmanager erfragt die fehlende Information auf Basis des Kontextes. Gibt der Anwender darauf keine Antwort, beendet sich das System (rechte gestrichelte Linie). Andernfalls wird das Fulfillment des Intents ausgeführt (technische Funktion und daher in der Abbildung nicht dargestellt) und dem Anwender eine Erfolgsmeldung ausgegeben.

Dieser Dialogbaum stellt einen einfachen Anwendungsfall dar. Solche Übersichten können auf unterschiedlichen Ebenen der Granularität in der Konzeption angelegt werden. Auf Anwendungsfallebene, Menüebene oder als Übersicht des gesamten Systems. Letzteres ist für ein Smart Home System mit vielen heterogenen Funktionen nicht umsetzbar.

5.3.3.5 Gesamtkonzeption

Für das vorliegende System ist eine gesamtheitliche Lösung durch ein Werkzeug aus den beschriebenen Gründen nicht möglich. Das Dialogsystem setzt sich insgesamt aus drei einzelnen Teilen zusammen, Pocketsphinx zur Offline Wakeword-Aktivierung, Cloud Speech-To-Text zur automatischen Spracherkennung und Dialogflow zur intelligenten Verarbeitung der Nutzereingaben als Dialogmanager. Für die Implementierung sind konzeptionelle Vorgaben der Aufrufreihenfolge und damit Steuerung der Prozesse sowie Öffnung der verschiedenen Übertragungskanäle wichtig.

In Abbildung 9 ist die Aufrufreihenfolge in einem Sequenzdiagramm veranschaulicht. Die Steuerung der einzelnen Prozesse soll durch die MainActivity des Smart Home Systems übernommen werden. Neben der Instanziierung der drei Elemente soll der Aufruf in weitere Prozesse unterteilt werden.

Der Prozess der Wakeword-Erkennung läuft permanent im Haupt-Thread (MainActivity), bis der Anwender das Schlüsselwort "My Assistant" spricht (Schritt 4). Wurde das Wakeword gesprochen, beendet sich der Prozess und ein neuer Prozessschritt wird innerhalb des Main-Threads eröffnet (Schritt 5).

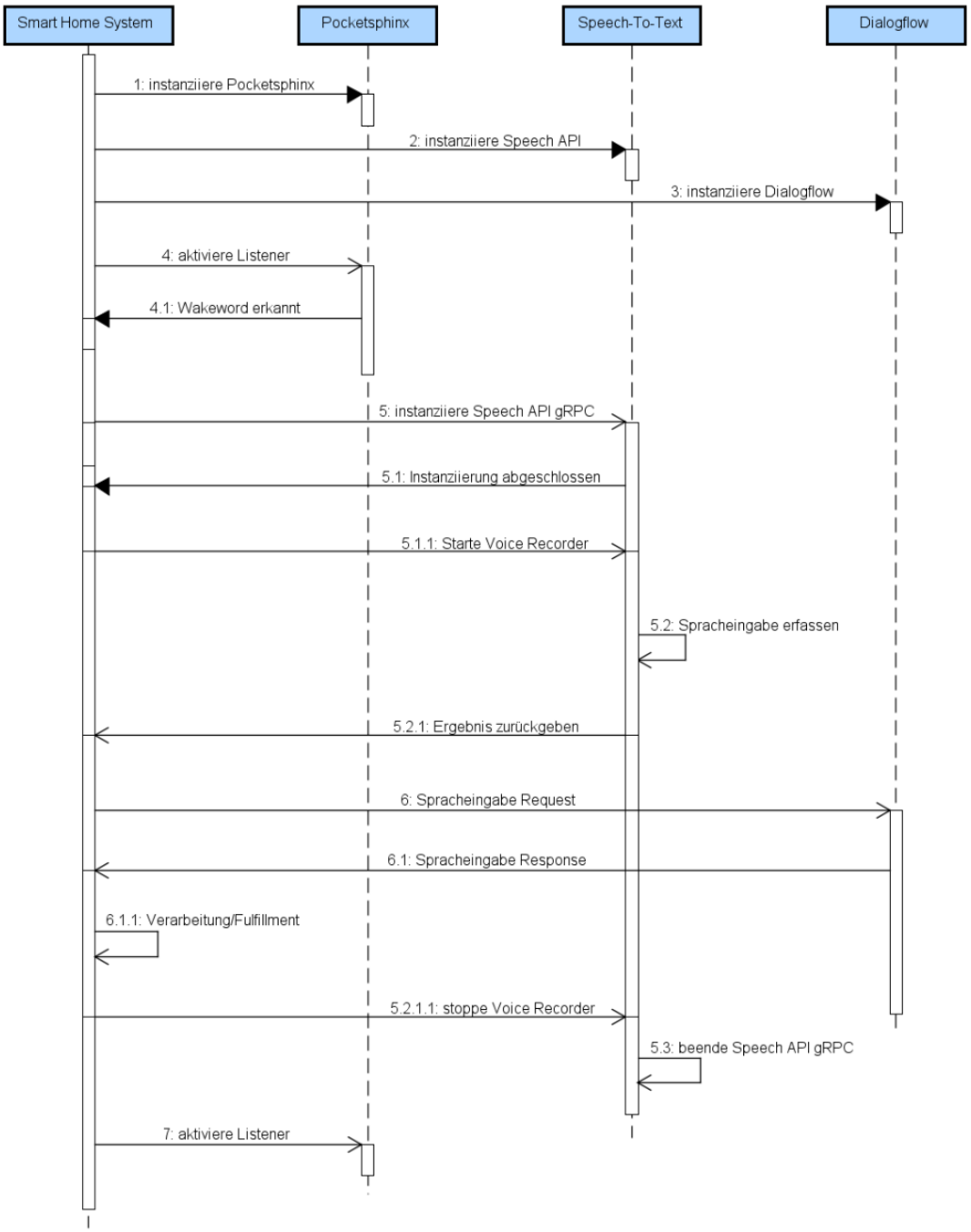

Abbildung 9: Sequenzdiagramm Dialogsystem

Das Handling dieses Prozessabschnittes ist für die Umsetzung als der komplexeste Schritt zu betrachten. Zum einen muss der Zugriff auf den gRPC-Aufruf initialisiert werden zum anderen der Voice Recorder als zentrales Element der Sprachaufzeichnung aktiviert werden. Die Erfassung der Spracheingaben wird vollständig durch den Cloud-Service ausgeführt (Schritt 5.2). Nach Rückgabe der transkribierten Sprachnachricht, darf der Prozess jedoch nicht geschlossen werden. Das Konzept sieht vor, dass nach der Verarbeitung der natürlichsprachlichen Eingaben durch den Dialogma-

nager (Schritt 6), der Kontext für weitere Fragen beibehalten wird. Das Beenden des Prozesses würde eine weitere Aktivierung durch das Wakeword voraussetzen. Unter Betrachtung der UX würde dies einen erheblichen Nachteil in der Bedienung des Systems wiederspiegeln. Folglich können die Schritte 5.1.1 bis 5.2.1 sowie 6 beliebig oft durchgeführt werden. Ein Beispiel hierfür ist in Abbildung 8 zu finden. Sofern dem Dialogmanager noch Informationen zur abschließenden Bearbeitung des Intents fehlen, darf der Voice Recorder nicht gestoppt werden.

Sind alle benötigten Informationen gesammelt und die Verarbeitung des Intents im Hauptprozess der Anwendung abgearbeitet, wird der Voice Recorder gestoppt. In diesem Zuge ist eine erneute Aktivierung des Pocketsphinx-Listeners (Schritt 7) erforderlich.

5.3.4 Begleitanwendung

Begleitanwendungen sind im Umfeld des IoT eine häufig eingesetzte Möglichkeit, um mit dem jeweiligen IoT-System dezentral kommunizieren zu können. Die Begleitanwendung, oder auch Companion App, wird mit dem Betriebssystem Android umgesetzt und als Smartphone-Applikation ausgeliefert.

Die Funktionen der Anwendung sind zum einen das Empfangen von Nachrichten, ausgelöst durch das Smart Home System, zum anderen der Zugriff und Manipulation der Daten des Smart Home Systems über eine Datenbank. Die Kopplung beider Anwendungen erfolgt über eine eindeutige ID, auf die sowohl Smart Home System als auch Companion App Zugriff haben. Die Übertragung aller Funktionen des Smart Home Systems auf die Smartphone-Anwendung ist nicht möglich, da beide Anwendungen unterschiedlichen Bibliotheken des Android Betriebssystems nutzen. Beispielsweise ist der Zugriff auf den I2C-Bus zur Übertragung der Sensordaten nur über den Android Things-Prototyp abbildbar.

5.3.5 Cloud Services

Die Cloud-Services Speech-To-Text und Dialogflow der Google Cloud Platform, wurden in den vergangen Kapitel genauer vorgestellt. Nun soll eine Funktionsbeschreibung weiterer Services hinsichtlich derer Funktion erfolgen.

Für die Verarbeitung der Sensordaten nach dem Publish/Subscribe-Paradigma sollen zwei Cloud-Funktionalitäten eingesetzt werden. Die beiden Funktionen Pub/Sub und BigQuery sind Werkzeuge zur automatischen Verarbeitung großer Datenmengen und lassen sich unter dem Schlagwort Big Data eingruppieren. Pub/Sub bietet die Möglichkeit Themen zu erstellen und Abonnements (Subscribe) zu Nachrichten (Publish) eines Themas anzulegen. Durch die asynchrone Nachrichtenübertragung, werden durch den Dienst Sender und Empfänger voneinander entkoppelt. Die Nachrichtenübertragung ist hochverfügbar und verschlüsselt. Zur Weiterverarbeitung ist die Anbindung an weitere Services möglich.[197]

Dazu zählt BigQuery, eine Cloud-basierte Data Warehouse-Lösung, die eine SQL-orientierte Managementoberfläche zur Analyse von großen Datenmengen bietet.[198] Im Umfang dieser Arbeit findet ein Export der Temperatur- und Luftdruckdaten des Pub/Sub-Services, über einen konfigurierten Job, in die BigQuery Datenbank, statt. Der technische Durchstich begrenzt den Funktionsumfang dieses Services im Rahmen des Prototyps.

Für das Zusammenspiel zwischen Companion App und Smart Home Prototyp, sollen zwei Funktionen des Firebase Cloud-Services genutzt werden. Diese sind erforderlich für die Umsetzung des Anwendungsfalls Begleitanwendung aus Abbildung 6. Das Versenden und Empfangen von Nachrichten erfolgt über Firebase Cloud Messaging. Vorteile dieses Services sind die einfache Integration in Funktionen der Google Cloud Platform sowie die Verfügbarkeit eines Android SDKs. Nachrichten können entweder dem gesamten Benutzerkreis einer Anwendung oder über eine ID, einem spezifischen Benutzer übermittelt werden.[199]

Die Firebase Realtime Database ist eine NoSQL-Cloud Datenbank zur Speicherung von dokumentbasierten Objekttypen. Zentrale Eigenschaften sind die Synchronisierung der Daten zwischen allen Knoten in Echtzeit sowie die virtuelle Verfügbarkeit, wenn ein Gerät offline ist. Dies gelingt durch temporäre Speicherung der Daten auf dem Gerät und automatische Synchronisierung, sobald wieder ein Netzwerkzugriff

[197] Vgl. Google Cloud o. V. (2018).
[198] Vgl. Google Cloud o. V. (o. J. b)
[199] Vgl. Firebase o. V. (2018a).

besteht.[200] Für den Anwendungsfall, der Einstellung der Raumtemperatur, ist aber die Echtzeit-Funktion und die Möglichkeit zur Implementierung eines Listeners, zum Abfragen des aktuellen Datenbankstatus auf Seiten des Clients, entscheidend. Darüber hinaus erleichtert die Eigenschaft der Schemafreiheit die Erstellung des Prototyps, bei der Erreichung der Abbildung eines vertikal vollständigen Anwendungssta-Anwendungsstapels.

5.4 Systemarchitektur

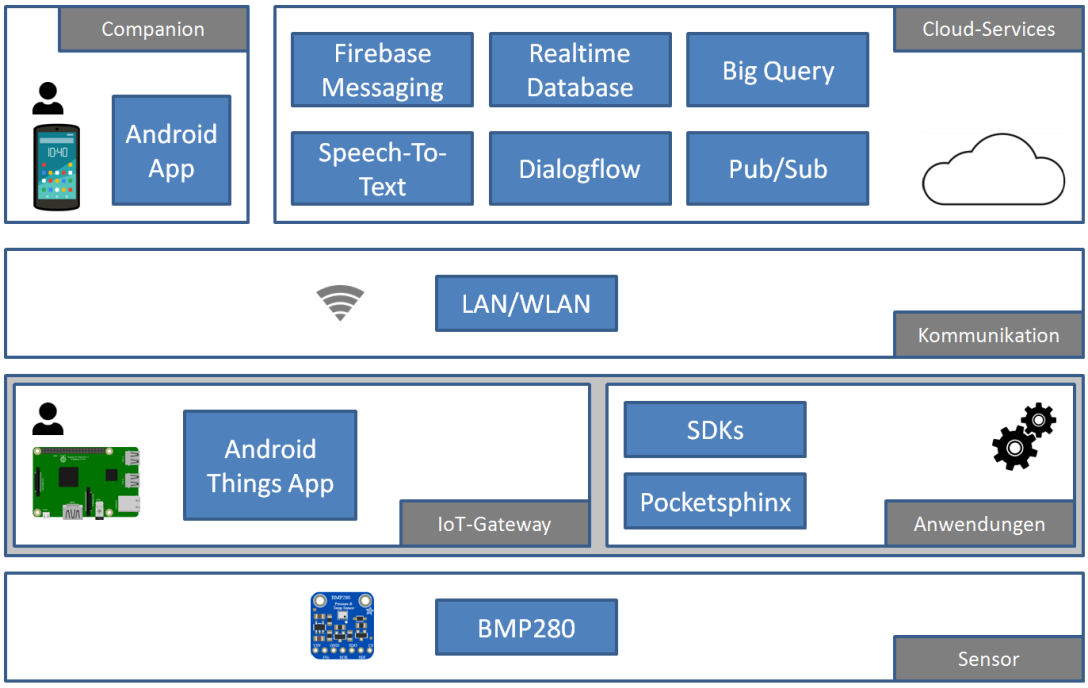

Abbildung 10: Systementwurf der einzelnen Komponenten

Die oben dargestellte Anwendungsarchitektur fasst auf grobgranularer Ebene die unterschiedlichen Technologien zusammen, die im Kontext der Entwicklung des Prototyps genutzt werden.

Die beschriebenen Systeme, Werkzeuge und Dienstleistungen bauen auf theoretischen Untersuchungen aus den Kapitel 2, 3 und 4 auf. Die Erkenntnisse der Voruntersuchungen wurden ebenso bei der Evaluation der eingesetzten Technologien berücksichtigt, wie die spezifischen Eigenschaften.

Bisher nicht explizit dargestellt wurde die Kommunikationsschicht. Da es sich bei dem Smart Home Prototyp, um ein Gerät handelt, welches ausschließlich zur An-

[200] Vgl. Firebase o. V. (2018b).

wendung im Haus konzipiert ist, haben neben der kabelgebundenen LAN- und der kabellosen WLAN-Schnittstelle, keine weiteren Netzwerke zur Datenübertragung eine größere Bedeutung. Die Nutzung der BLE-Schnittstelle des Raspberry sowie die Implementierung dieser auf Seiten des Betriebssystems, ist für die Ausgabe der Signale und Sprachantworten, über verteilte Lautsprechersysteme interessant, findet jedoch keine weitere Betrachtung. Aufgrund der Konzeption nicht weiter berücksichtigt ist die mobile Nutzung des Systems, da diese Funktion sinnvollerweise durch Smartphones oder Wearables abgedeckt wird.

6 Prototypische Entwicklung des Smart Home Systems

6.1 Einleitung

In diesem Kapitel wird die Entwicklung der aus dem Konzept stammenden Vorgaben, verknüpft mit den theoretischen Voruntersuchungen der Kapitel 2 - 5, beschrieben.

Eine vollständige Darstellung des Quellcodes ist im Rahmen dieser Arbeit nicht möglich und an vielen Stellen nicht erforderlich. Die wichtigen Auszüge der Funktionen und Anwendungsfälle, sind als Abbildungen des Quellcodes in den Fließtext eingebunden. Ergänzende Methoden und Konfigurationseinstellungen sind im Anhang aufgelistet. Ein Verweis auf diese Quellen findet jeweils in der Fußnote statt.

Zur besseren Übersicht, an welcher Position des Systems sich die beschriebenen Umsetzungsbeispiele befinden, hilft zum einen die Systemarchitektur (Abbildung 10). Zum anderen ist das Sequenzdiagramm aus Kapitel 5.3.3.5 mit den entsprechenden textuellen Erläuterungen zum Verständnis des Quellcodes und Reihenfolge der nebenläufigen Prozesse, hilfreich.

Abschließend liefert das auf der nächsten Seite folgende Klassendiagramm eine Zusammenfassung der wichtigsten Klassen, Attribute und Methoden sowie deren Beziehung untereinander. Zum Verständnis verschiedener Aufrufreihenfolgen und zur Einordnung der implementierten Funktionen ist dies sowohl für die Umsetzung als auch für das nachträgliche Verständnis essenziell. Aus Gründen der Übersichtlichkeit ist das Klassendiagramm nicht vollständig mit allen Attributen und Methoden aufgelistet. Alle wichtigen Komponenten jedoch, welche in der Umsetzung referenziert werden, sind abgebildet.

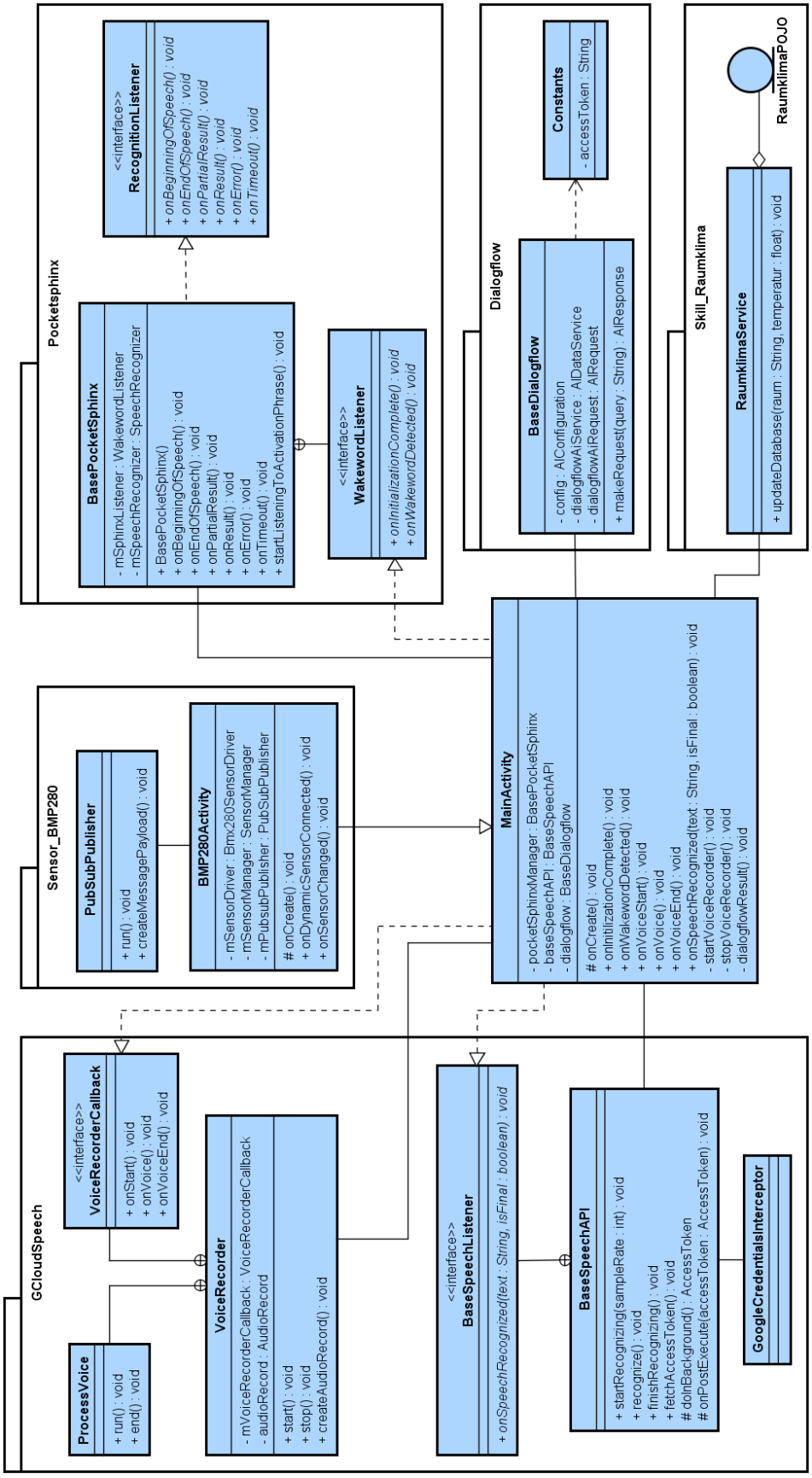

Abbildung 11: Klassendiagramm des Smart Home System

6.2 Grundlagen

In Kapitel 5.3.2 sind die Eigenschaften des Betriebssystems Android Things beschrieben. Auf einige dieser Eigenschaften wird in diesem Abschnitt, mit dem Kontext des Smart Home Systems, eigegangen. Zudem werden verschiedene Grundlagen des Android Things-Prototyps vorgestellt, auf welche sich die nachfolgenden Kapitel beziehen.

Für die Entwicklung einer Android-Anwendung ist die Android Debug Bridge (ADB) ein wichtiges Werkzeug. Die ADB ist ein Kommandozeilen-Werkzeug mit dem die Verbindung zu einem Entwicklungsgerät (z.B. RaspberryPi oder Smartphone), über das Netzwerk oder per USB, hergestellt werden kann. Eingebunden in eine beliebige Betriebssystem-Shell, können so Anwendungen installiert, Datenspeicherzugriffe ausgeführt oder erweiterte Debugging-Ausgaben erzielt werden.

Die in einer Android Things-Anwendung genutzten Activities, Rechte und Services werden in der Manifest.xml-Datei eingetragen.[201]

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="com.google.android.things.permission.MODIFY_SCREEN_SETTINGS" />
<uses-permission android:name="com.google.android.things.permission.MANAGE_INPUT_DRIVERS" />
<uses-permission android:name="com.google.android.things.permission.MANAGE_SENSOR_DRIVERS" />
```

Abbildung 12: Auszug Berechtigungen der Manifest.xml

Für die Smart Home Anwendung sind verschiedene Standardberechtigungen von Android erforderlich. Diese sind dem Namen nach selbsterklärend und werden nicht weiter betrachtet. Spezielle Berechtigungen der IoT-Erweiterung von Android, betreffen die Ausrichtung des Bildschirms (MODIFY_SCREEN_SETTINGS) und die Nutzung sowie den Zugriff auf externe Sensoren (MANAGE_INPUT_DRIVERS, MANAGE_SENSOR_DRIVERS). Darüber hinaus wird der Activity, welche den Einstieg in die Anwendung darstellt, eine weitere Kategorie zugeordnet (IOT_LAUNCHER). Da Anwendungen des IoT sind nicht an eine grafische Oberfläche gebunden sind, wird so der Start der Anwendung festgelegt.

---

[201] Vgl. Anhang A3.1.1.

In der Umsetzung werden viele verschiedene Werkzeuge eingesetzt, die in der Konzeption des Systems bereits vorgestellt wurden. Abhängigkeiten der genutzten Bibliotheken werden über die build.gradle-Datei der Anwendung integriert. Im Anhang A3.1.2 befindet sich ein Auszug aller genutzten Abhängigkeiten. Spezielle Eigenschaften dieser Bibliotheken werden im Rahmen der Umsetzung weiter erläutert.

Die MainActivity ist zentraler Knotenpunkt der IoT-Anwendung. Über diese Klasse werden die im Anschluss vorgestellten Funktionen instanziiert, initialisiert und zum Teil auch in ihren Prozessen gesteuert. Einstiegspunkt der Anwendung ist die onCreate-Methode der MainActivity. Hier werden grundlegende Layout-Eigenschaften definiert, sowie die Klassen BasePocketsphinx, BaseSpeechAPI und TextToSpeech instanziiert. Des Weiteren wird die BMP280Activity gestartet.

6.3 Umsetzung

6.3.1 Funktion Wakeword

Alle notwendigen Abhängigkeiten werden über das Pocketsphinx Android-Archiv bereitgestellt. Folglich sind neben der Integration dieses Archivs und der Sprachmodelle, keine weiteren Abhängigkeiten in der Gradle-Datei der Anwendung anzugeben. Basis für die Entwicklung der Funktion bildet die Demoanwendung für mobile Android-Systeme von CMUSphinx.[202]

Die Klasse BasePocketSphinx repräsentiert in der Anwendung des Smart Home Prototyps die Basisklasse des Toolkits. Sie implementiert einen RecognitionListener, der aus der abhängigen Bibliothek von Pocketsphinx stammt. Das Interface beinhaltet die Methoden onBeginningOfSpeech, onEndOfSpeech, onPartialResult, onResult, onError und onTimeout.[203] Diese Methoden fungieren als Callback-Funktionen, um zur Laufzeit auf bestimmte Ereignisse des SpeechRecognizers reagieren zu können. Die beiden interessantesten Methoden sind onPartialResult und onResult. Diese werden aufgerufen, wenn Spracheingaben getroffen werden und prüfen, ob das Wakeword erkannt, die Spracheingabe beendet ist und der SpeechRecognizer gestoppt werden soll. Das Hypothesis-Objekt der Spracheingabe enthält neben dem transkri-

[202] Vgl. CMUSphinx o.V. (o. J. e).
[203] Vgl. Anhang A3.2.

bierten Text als String, auch Informationen über die Wahrscheinlichkeit, dass der ausgegebene Text dem Eingangssprachsignal entspricht.

Der SpeechRecognizer wird bei der Instanziierung der BasePocketSphinx-Klasse aus der MainActivity erzeugt und mittels asynchronem Prozessaufruf initialisiert (vgl. Abbildung 13).

```
public BasePocketSphinx(final Context context, final WakewordListener mSphinxListener) {
    this.mSphinxListener = mSphinxListener;

    new AsyncTask<Void, Void, Exception>() {
        @Override
        protected Exception doInBackground(Void... params) {
            try {
                Assets assets = new Assets(context);
                File assetsDir = assets.syncAssets();

                mSpeechRecognizer = SpeechRecognizerSetup.defaultSetup()
                        .setAcousticModel(new File(assetsDir, child: "en-us-ptm"))
                        .setDictionary(new File(assetsDir, child: "cmudict-en-us.dict"))
                        .getRecognizer();

                mSpeechRecognizer.addListener(BasePocketSphinx.this);

                //Wakeword definieren
                mSpeechRecognizer.addKeyphraseSearch(WAKEUP_SEARCH, WAKEWORD);
            } catch (IOException e) {
                return e;
            }
            return null;
        }

        @Override
        protected void onPostExecute(Exception result) {
            if (result != null) {
                Log.e(TAG, msg: "Failed to initialize recognizer: " + result);
            } else {
                mSphinxListener.onInitializationComplete();
            }
        }
    }.execute();
}
```

Abbildung 13: Instanziierung und Konfiguration des SpeechRecognizers

Die Konfiguration des SpeechRecognizers wird in der Methode doInBackground zur Laufzeit der MainActivity durchgeführt. Die Dateien der Modelle zur Verarbeitung der Sprachsignale sind als Assets im Anwendungskontext, in einer XML-Datei hinterlegt. Der oben dargestellte Aufbau zeigt, dass ein englisches acoustic model sowie ein englisches Wörterbuch für die Verarbeitung genutzt werden. Des Weiteren wird dem SpeechRecognizer der Listener der BasePocketSphinx-Klasse zugewiesen, um auf Eingaben dieses spezifischen SpeechRecognizer-Objekts reagieren zu können. Die Konstante WAKEWORD ist ein einfacher String, der das im Konzept festgeleg-

te Schlüsselwort "My Assistant" enthält und der Aktivierung des im nächsten Kapitel beschriebenen ASR-Systems dient.

```
@Override
public void onInitializationComplete() {
    Log.d(TAG,  msg: "Pocketsphinx Speech Recognizer successfully activated");
    pocketSphinxManager.startListeningToActivationPhrase();
}

@Override
public void onWakewordDetected() {
    Log.d(TAG,  msg: "Wakeword detected -> My Assistant");
    MediaPlayer mediaPlayer = MediaPlayer.create( context: this, R.raw.activation_notification);
    mediaPlayer.start();
    //Start Cloud Speech-To-Text VoiceRecorder
    startVoiceRecorder();
}
```

Abbildung 14: Implementierung des WakewordListeners in der MainActivity

In Abbildung 14 sind die Quellcodeauszüge, der in der MainActivity implementierten Methoden, des WakewordListeners dargestellt. Sobald die Initialisierung des SpeechRecognizers im beschriebenen, asynchronen Task, abgeschlossen ist, wird die Methode onInitializationComplete gerufen. Wird das Wakeword erkannt, folgt der Aufruf der onWakewordDetected Listener-Methode. Diese gibt zum einen ein akustisches Signal für den Anwender aus zum anderen aktiviert sie den VoiceRecorder der Cloud Speech-To-Text Spracherkennung.

6.3.2 Funktion Spracherkennung

Basis jeder Funktionalität der Google Cloud Platform ist ein Projekt, welches über die Konsole der Web-Anwendung angelegt werden kann. Jedes Projekt besitzt eine ID sowie ein spezifisches Identity & Access Management (IAM). Über dieses werden Schreib- und Leserechte vergeben und die für die Entwicklung der Anwendung benötigten API-Funktionalitäten aktiviert.

Die Verknüpfung zwischen Anwendung und Projekt erfolgt über einen Access Token, der im JSON-Format in die Anwendung integriert wird. In Anhang A3.1.2 sind die grundlegenden Abhängigkeiten der Gradle-Konfiguration, die für die Autorisierung der Anwendung über Open Authorization (OAuth) und für die Entwicklung mit der Cloud Speech API erforderlich sind, dargestellt. Daneben gibt es noch weitere Abhängigkeiten, wie die Remote Procedure Calls (gRPC) zum Aufruf der Cloud Funktionen und generelle Abhängigkeiten der Android Laufzeitumgebung. Die Prüfung des Access Tokens soll an dieser Stelle nicht detailliert betrachtet werden. Ein-

fach dargestellt, erfolgt diese in einem asynchronen Aufruf[204] der Klasse BaseSpeechAPI, welche über die GoogleCredentialsInterceptor-Klasse via HTTP, im Erfolgsfall, einen gRPC-Kanal zur Anwendung aufbaut. Die Prüfung umfasst dabei auch die Validität des in der JSON-Datei hinterlegten Schlüssels zur Projekt-ID sowie das Ablaufdatum des Tokens. Bei erfolgreicher Prüfung wird der gRPC-Kanal zum Aufruf der Cloud Speech API freigegeben.

```
private final VoiceRecorder.VoiceRecorderCallback mVoiceVoiceRecorderCallback =
        new VoiceRecorder.VoiceRecorderCallback() {

            @Override
            public void onVoiceStart() {
                if (baseSpeechAPI != null) {
                    baseSpeechAPI.startRecognizing(mVoiceRecorder.getSampleRate());
                }
            }

            @Override
            public void onVoice(byte[] data, int size) {
                if (baseSpeechAPI != null) {
                    baseSpeechAPI.recognize(data, size);
                }
            }

            @Override
            public void onVoiceEnd() {
                if (baseSpeechAPI != null) {
                    baseSpeechAPI.finishRecognizing();
                    stopVoiceRecorder();
                    pocketSphinxManager.startListeningToActivationPhrase();
                }
            }

        };
```

Abbildung 15: Implementierung des VoiceRecorderCallback-Interfaces in der MainActivity[205]

Relevant für die Umsetzung der funktionalen Anforderungen ist der Aufbau des VoiceRecorders. Dieser wird in der MainActivity durch Aufruf der Methode startRecognizer, aus der onWakewordDetected-Methode des Pocketsphinx-Listeners, instanziiert.[206] Als Übergabeparameter erhält dieser, das in der MainActivity implementierte VoiceRecorderCallback-Interface, zur Steuerung des VoiceRecorders im Hauptprozess der Anwendung. Dieses Interface besitzt die Methoden onVoiceStart, onVoice und onVoiceEnd (vgl. Implementierung in Abbildung 15).

---

[204] Vgl. Anhang A3.3.1.
[205] in Anlehnung an: Thumar (2017).
[206] Vgl. Anhang A3.3.2.

In der Methode onVoiceStart wird die Konfiguration sowie das Request-Objekt für die Übertragung der transkribierten Daten des Speech-To-Text Cloud-Services erstellt. Die Konfiguration des Recognizers beinhaltet Informationen über das Audio-Encoding sowie die Sprache, die das ASR-System verarbeiten soll. Für den Smart Home Prototyp ist die deutsche Sprache Basis des Systems.[207]

Die Methode onVoice adressiert einen parallel laufenden Prozess, welcher die Sprachsignale des Anwenders aufnimmt und an die Cloud Speech API zur Verarbeitung sendet. Dieser asynchron ausgeführte Prozess, wird beim Start des VoiceRecorders kreiert. Während des Streamingprozesses (onVoice), wird dem BaseSpeechAPI-Listener der transkribierte Text sowie der aktuelle Status zugewiesen. Der Listener ist wiederum in der MainActivity implementiert (vgl. Abbildung 16). Die Parameter der Listener-Methode onSpeechRecognized, sind der Text in Form eines Strings sowie ein Statuskennzeichen, ob der Verarbeitungsprozess abgeschlossen wurde. Ist der Verarbeitungsprozess abgeschlossen, wird die nächste Prozessstufe, des in Abschnitt 5.3.3.5 konzipierten Ablaufs, aufgerufen (dialogflowResult).

```
private final BaseSpeechAPI.Listener mSpeechServiceListener =
        new BaseSpeechAPI.Listener() {
            @Override
            public void onSpeechRecognized(final String text, final boolean isFinal) {

                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        if (isFinal) {
                            //Show transcribed message
                            makeText(getApplicationContext(), text, Toast.LENGTH_SHORT).show();
                            //Send transcribed message to Dialogflow-Agent
                            dialogflowResult(dialogflow.makeRequest(text));
                        } else {
                            //Show live streaming results while processing
                        }
                    }
                });
            }
        };
```

Abbildung 16: Prozesshandling für die transkribierte Spracheingabe in der MainActivity[208]

Am Ende des Sprachsignals wird die Methode voiceEnd ausgeführt. Der Streaming-Kanal wird geschlossen und dem in der Konfiguration erzeugten StreamingRecognizeRequest-Objekts wird die Referenz "null" zugewiesen. Ab-

[207] Vgl. Anhang A3.3.3.
[208] in Anlehnung an: Thumar (2017).

schließend erfolgt die erneute Aktivierung des Wakeword-Listeners durch Aufruf der Methode startListeningToActivationPhrase der BasePocketSphinx-Klasse.

6.3.3 Funktion Dialogverarbeitung

6.3.3.1 Dialogdesign

Die Funktionsweise von Dialogflow ist in den Kapiteln Dialogverarbeitung - Dialogflow und Dialogdesign des Konzepts bereits beschrieben worden. Hier soll nun exemplarisch dargestellt werden, wie die Umsetzung von Intents, Entities und Trainingsphrasen in der Praxis funktioniert. Als Beispiel wird der Anwendungsfall Temperatursteuerung auf Seiten des Smart Home Prototyps vorgestellt. Die Einstellung der Temperatur soll per Spracheingabe erfolgen und eine spezifische Behandlung einzelner Räume und deren Temperaturen ermöglichen.

Ein Dialogflow-Projekt ist Grundvoraussetzung für den Beginn der Dialogumsetzung. Die Konfiguration der Eigenschaften z.B. zur Sprachwahl oder Einbindung bereits vorhandener Prebuilt-Agents ist dem Entwickler überlassen. Für das Smart Home System ist die Sprache Deutsch als Standard und Englisch als ergänzende Sprache in der Konsole der Webanwendung konfiguriert.[209] Das erstellte Projekt, kann mit einem Google Cloud Projekt verbunden werden, sodass Funktionalitäten der Cloud Platform, zur Erfüllung der im Hintergrund auszuführenden Aufgaben bspw. als Web-Service, angebunden werden können.

Neben der Projektkonfiguration muss bestimmt werden, was der Dialogmanager für Dialogsituationen bedienen soll. Hierfür werden Intents definiert. Um eine Dialogsituation auf Augenhöhe zu schaffen, wie in den theoretischen Grundlagen der Spracherkennung beschrieben, muss auch auf Anfragen, die keinem Intent zugordnet werden können, eine Antwort gegeben werden. Abgedeckt wird dies durch Default- oder Fallback-Intents. Diese definieren standardisierte Ausgaben indem sie dem Anwender mitteilen, dass z.B. eine Frage nicht beantwortet werden kann oder verweisen auf eine Hilfe.

---

[209] Vgl. Anhang A2.1.1.

Beim Anwendungsfall der Raumtemperatursteuerung, wird ein TemperaturIntent benötigt. Dieser Intent hat zwei Parameter, die zur vollständigen Erfüllung des Anwendungsfalls benötigt werden - die Temperatur und der Raum.

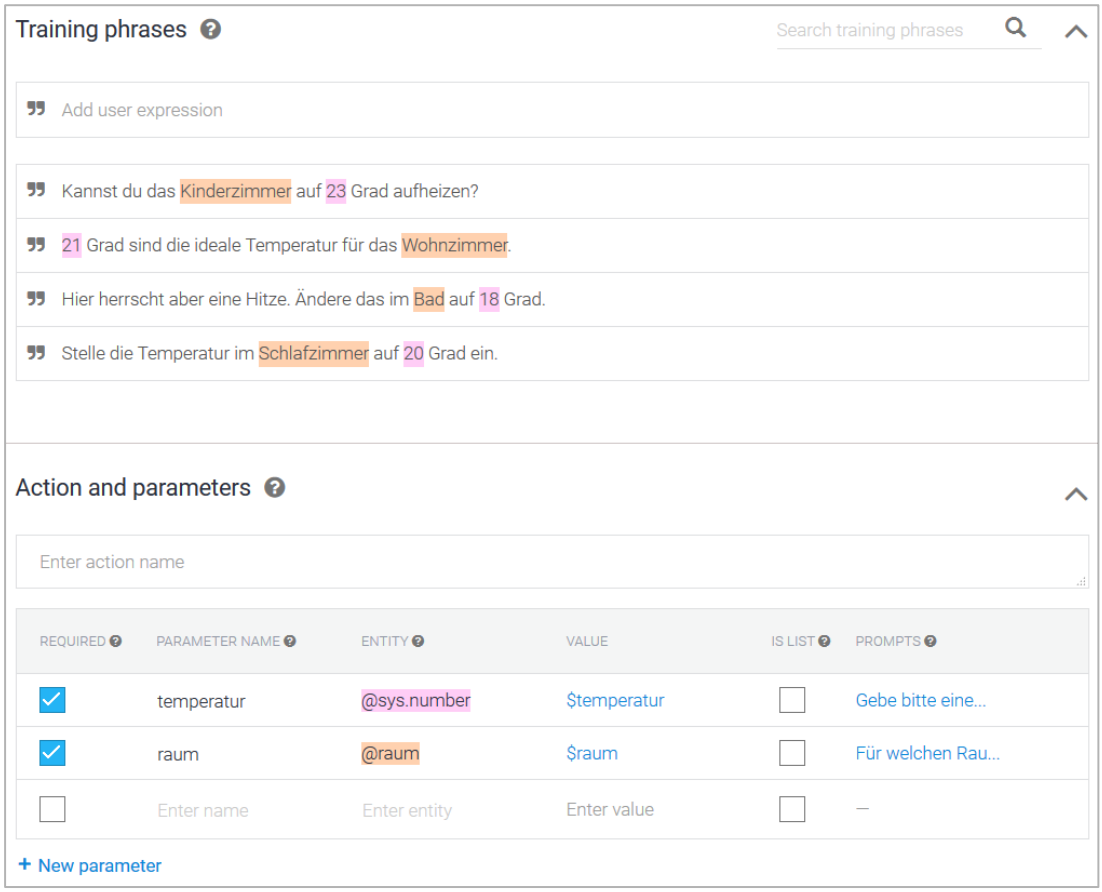

Abbildung 17: TemperaturIntent Training phrases und parameters

Diese beiden Parameter sind in der oben dargestellten Abbildung orange (Raum) und rosa (Temperatur) aufgeführt. Unter dem Reiter "Action and parameters" wird deutlich, dass beide Parameter zur Erfüllung des Auftrags benötigt werden. Im oberen Reiter "Training phrases", sind verschiedene Sätze definiert, die vom Anwender zur Aktivierung dieses Intents genutzt werden könnten. Die Machine Learning-Algorithmen von Dialogflow erhalten dadurch, neben den beispielhaften Sätzen, auch eine genaue Zuordnungen von Textteilen zu festen Parametern. Kinderzimmer und Wohnzimmer sind dem Parameter Raum zuzuordnen, 23 und 21 dem Parameter Temperatur.

Diese beiden Parameter sind die Entities dieses Intents. Der grundlegende Unterschied zwischen beiden ist, dass der Parameter Temperatur einer System-Entity (sys.number) zugeordnet ist, die Zahlen einer Spracheingabe identifiziert. Die Entity

des Parameters Raum, ist ein durch den Entwickler bereitgestellter Baustein. In Abbildung 18, ist die Umsetzung dieser Entity visualisiert. Zu einzelnen Einträgen, die zur Laufzeit genau diesem Parameter zugeordnet werden, gibt es bei Räumen verschiedene Synonyme. Durch die Angabe dieser Synonyme, kann die Treffergenauigkeit erhöht und damit die Benutzererfahrung verbessert werden.

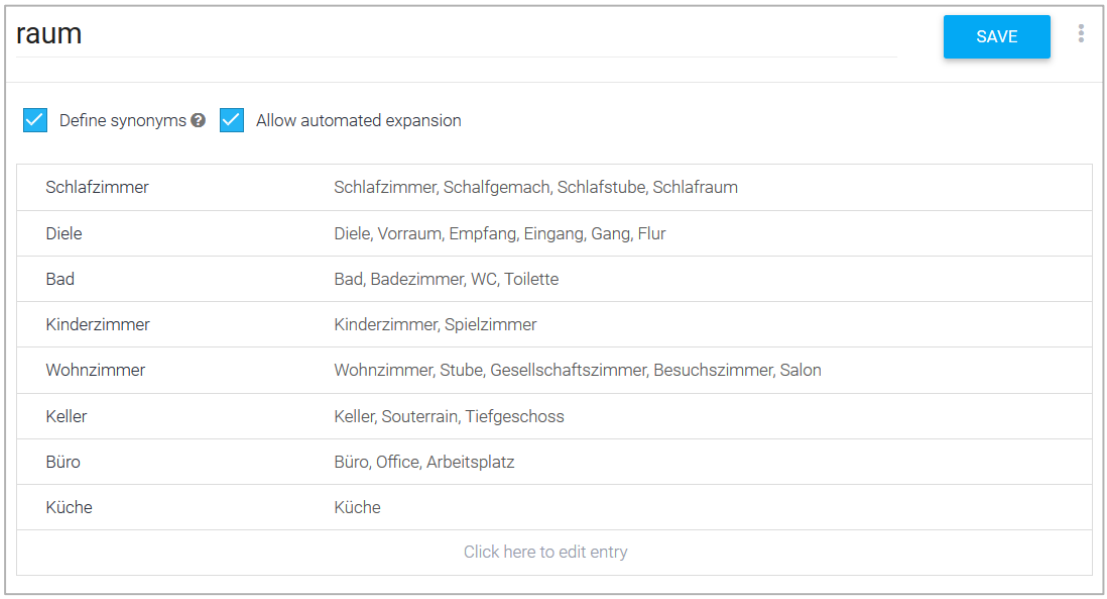

Abbildung 18: Umsetzung der Entity Raum

Zusätzlich können zur Verbesserung der Erkennung von Intents, in der Konsole der Webanwendung, vollständige Dialoge getestet werden. Die textuell eingegebenen Sätze werden durch den Dialogmanager bearbeitet und das Ergebnis, welches der Anwender erhält, samt identifizierten Parameter und deren Werte, ausgegeben.[210]

Die Formulierung der Antwort an den Nutzer, wird im Intent festgelegt.[211] Auch hier ist es möglich verschiedene Antworten zu definieren, damit eine abwechslungsreiche Dialogsituation geschaffen wird.

6.3.3.2 Implementierung

Die Implementierung des Dialogflow-Managers auf dem Smart Home System, zur Hintergrundverarbeitung der Intents, erfolgt mit Hilfe des Dialogflow SDKs für Android. Die benötigten Abhängigkeiten werden in der Gradle-Datei der Anwendung aufgenommen. Darüber hinaus ist auch hier ein Token zur Identifizierung des Ent-

[210] Vgl. Anhang A2.1.2.
[211] Vgl. Anhang A2.1.3.

wicklers und des Projekts notwendig. Dieser Token wird im Kontext der prototypischen Umsetzung als Konstante in die Anwendung integriert.

Für den Aufruf der Dialogverarbeitung ist nur eine Methode notwendig, welche den zuvor, durch die Funktion Spracherkennung, gewonnen Eingabewert als String an den Dialogflow-Request bindet (makeRequest). In dem unten dargestellten Quellcodeauszug wird die Konfiguration des Dialogmanagers sowie die Instanziierung des Daten-Services und Request-Objekts aufgeführt. Die unterstützte Sprache und das ASR-System sind hier als "German" und "System" angegeben. Da Android Things kein Betriebssystem-internes System zur Spracherkennung besitzt, sind die dort angegebenen Werte für die Funktion des Prototyps irrelevant und werden über die Funktion Spracherkennung ersetzt. Der Request selbst, ist als asynchroner Aufruf in der Methode makeRequest implementiert. Der Rückgabewert der Methode ist ein AIResponse-Objekt, welches verschiedene Werte wie den Antworttext, aber auch die Parameterwerte der Entities enthält.

```
private final static AIConfiguration config = new AIConfiguration(
        Constants.DIALOGFLOW_ACCESS_TOKEN,
        AIConfiguration.SupportedLanguages.German,
        AIConfiguration.RecognitionEngine.System);

private final static AIDataService dialogflowAiService = new AIDataService(config);
private final static AIRequest dialogflowAiRequest = new AIRequest();

public AIResponse makeRequest(final String query) {
    try {
        return (AsyncTask) (params) → {
                dialogflowAiRequest.setQuery(query);
                try {
                    return dialogflowAiService.request(dialogflowAiRequest);
                } catch (AIServiceException e) {
                    e.printStackTrace();
                    return null;
                }
        }.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR).get();
    } catch (InterruptedException | ExecutionException e) {
        e.printStackTrace();
        return null;
    }
}
```

Abbildung 19: Konfiguration und Aufruf des Dialogflow-Dialogmanagers

Instanziiert wird die beschriebene Klasse in der MainActivity. Dort wird auch die makeRequest-Methode aufgerufen. In der, an gleicher Stelle implementierten, BaseSpeechAPI Listener-Methode onSpeechRecognized, wird sobald die Eingabe des Anwenders abgeschlossen und vollständig in einen String übersetzt ist, diese Methode aufgerufen (vgl. hierzu Abbildung 16).

Bisher findet noch keinerlei Interaktion mit dem Nutzer statt. Damit dem Anwender eine qualifizierte Antwort zur durchgeführten Einstellung der Raumtemperatur gegeben werden kann, muss die Antwort im AIResponse-Objekt des Dialogmanagers, noch mit Informationen angereichert oder weitere Funktionen ausgeführt werden. Für diesen Anwendungsfall ist eine Einstellung des vernetzten Heizungscontrollers über ein IoT-Protokoll z.B. MQTT sowie die Ablage der Daten in einer Datenbank, bspw. für den dezentralen Zugriff der Daten durch die Begleitanwendung, notwendig. Dieser Schritt wird in der Methode dialogflowResult in der MainActivity ausgeführt. Die Methode wird ebenfalls nach Abschluss der Nutzereingaben im BaseSpeechAPI-Listener aufgerufen, sobald die Methode makeRequest eine Antwort liefert. Das Handling für den vorgestellten Anwendungsfall ist unten dargestellt.

```
final HashMap<String, JsonElement> params = result.getParameters();

//Fulfilment des Intents Temperatur
if ((metadata != null ? metadata.getIntentName() : null) != null &&
        Constants.INTENT_TEMPERATUR.equals(metadata.getIntentName())) {

    if (params.get(Constants.PARAM_RAUM).getAsString() != null) {

        //Update Realtime Database via Service
        raumklimaService.updateDatabase(
                params.get(Constants.PARAM_RAUM).getAsString(),
                params.get(Constants.PARAM_TEMPERATUR).getAsFloat());

        //TODO: Send data to specific Heating Controller for Smart Home via MQTT

        final String speech = result.getFulfillment().getSpeech();

        Log.i(TAG, msg: "Speech: " + speech);
        tts.speak(speech, TextToSpeech.QUEUE_FLUSH, params: null);
    } else {
        tts.speak( text: "Leider ist hier ein Fehler unterlaufen. " +
                "Bitte versuche es neu.", TextToSpeech.QUEUE_FLUSH, params: null);
    }

} else {
    final String speech = result.getFulfillment().getSpeech();
    Log.i(TAG, msg: "Speech: " + speech);
    tts.speak(speech, TextToSpeech.QUEUE_FLUSH, params: null);
}
```

Abbildung 20: Fulfillment des TemperaturIntents in der MainActivity

Die Verarbeitung des AIResponse-Objekts erfolgt asynchron. Wie bereits angemerkt, beinhaltet das Antwort-Objekt verschiedene Nutzungsdaten, Metadaten und Parameter. Für die Ausführung des Anwendungsfalls wird der zurückgelieferte Intent, auf Übereinstimmung mit der Konstanten des Anwendungsfalls der Anwendung, geprüft. Anschließend werden die einzelnen Funktionen aufgerufen. In diesem Beispiel

die Methode updateDatabase, welche im Kapitel Anwendungsfall Begleitanwendung noch genauer betrachtet wird. Der Methode werden die Parameter Raum und Temperatur, aus dem Response-Objekt zur Speicherung übergeben. Die Anbindung an einen Heizungscontroller ist an dieser Stelle außer Acht gelassen. Zum Abschluss des Fulfillments, wird per Betriebssystem-internen Text-To-Speech, die Antwort des Dialogmanagers akustisch ausgegeben. Eine erneute Angabe der Parameter ist an dieser Stelle nicht erforderlich, da diese in der modellierten Antwort von Dialogflow integriert sind (vgl. hierzu auch A2.1.3).

6.3.4 Anwendungsfall Sensordaten

6.3.4.1 Grundlagen

Für die Umsetzung und Nutzung des BMP280 Mikrocontrollers, sind verschiedene Aspekte zu berücksichtigen. Im Rahmen der Erstellung des Prototyps wird die Integration des Sensors bspw. in das Gehäuse eines fertigen Produkts nicht betrachtet. Der Sensor wird mit so genannten Breakout pins verlötet, damit dieser in einer Steckplatine (oder auch: Breadboard) mit einer einfachen Verkabelung genutzt werden kann. Visualisiert ist dies in der unten stehenden schematischen Abbildung.

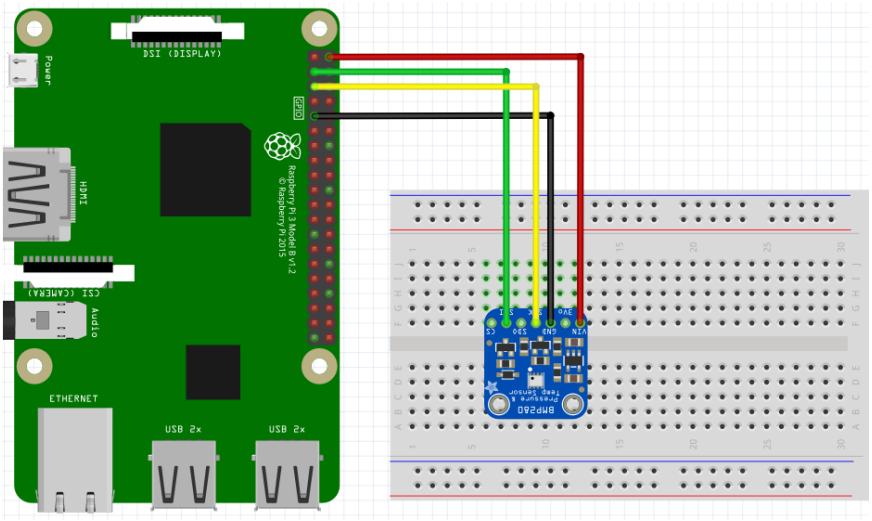

Abbildung 21: Verbindung Raspberry mit BMP280 Sensor

Der Mikrocontroller hat insgesamt sechs Anschlüssel für Breakout-Pins. Zwei dieser Anschlüsse werden nicht verwendet. Das ist zum einen ein zweiter Datenanschluss und zum anderen ein Anschluss zur Verbindung eines Displays, zur direkten Daten-

übertragung auf der Steckverbindung. Die Folgenden Anschlüsse werden für die Nutzung des Mikrocontrollers benötigt.

- Stromversorgung 5V, z.B. via GPIO-Pin 2
- Masse (Ground, GND), z.B. via GPIO-Pin 9
- Taktung (Serial Clock, SCK), z.B. via GPIO-Pin 5
- Datenübertragung (Serial Data In, SDI), z.B. via GPIO-Pin 3

6.3.4.2 Implementierung

Bei der Umsetzung dieses Anwendungsfalls zeigt sich ein Vorteil des Betriebssystems Android Things. Die benötigten Treiber zur Adressierung des Mikrocontrollers werden bereitgestellt und können als Abhängigkeit in die Gradle-Datei der Anwendung eingetragen werden. Basis dieser Implementierung ist die von Android Things bereitgestellte Beispielanwendung zum Anbinden des Sensortreibers und Auslesen der Daten.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    mSensorManager = ((SensorManager) getSystemService(SENSOR_SERVICE));

    try {
        mEnvironmentalSensorDriver = new Bmx280SensorDriver(BoardDefaults.getI2cBus());
        mSensorManager.registerDynamicSensorCallback(mDynamicSensorCallback);
        mEnvironmentalSensorDriver.registerTemperatureSensor();
        mEnvironmentalSensorDriver.registerPressureSensor();
        Log.d(TAG,  msg: "Initialized I2C BMP280");
    } catch (IOException e) {
        throw new RuntimeException("Error initializing BMP280", e);
    }

    //Cloud PubSub Publisher if credentials are present
    int credentialId = getResources().getIdentifier( name: "credentials",
            defType: "raw", getPackageName());
    if (credentialId != 0) {
        try {
            mPubsubPublisher = new PubSubPublisher( context: this,
                    appname: "masterThesisAssistant",
                    project: "masterThesisAssistant",  topic: "raumtemperatur", credentialId);
            mPubsubPublisher.start();
        } catch (IOException e) {
            Log.e(TAG,  msg: "error creating pubsub publisher", e);
        }
    }
}
```

Abbildung 22: Initialisierung des Sensors und Instanziierung der Cloud PubSub-Funktion[212]

[212] in Anlehnung an: Android Things o. V. (2018).

Der Anwendungsfall zum Auslesen der Temperatur- und Sensordaten ist in eine eigene Activity, TemperatureSensorActivity, ausgelagert. Sie ist aber als Kind-Klasse in der Vererbungshierarchie der MainActivity implementiert. Die Activity hat zwei wichtige Objekte deren Einsatz im Folgenden genauer betrachtet wird. Auf der einen Seite ist es das Treiber-Objekt des BMP280 Sensors (Bmx280SensorDriver) und auf der anderen Seite das SensorManager-Objekt, zur Registrierung des externen Sensors am Betriebssystem.

Der SensorManager wird über die aus dem Context-Objekt stammende Methode getSystemService initialisiert (vgl. Implementierung in Abbildung 22). Der Bmx280SensorDriver benötigt bei der Instanziierung einen Bus gegen den die Verbindung hergestellt wird. Der String hinter der Funktion getI2CBus ist für den RaspberryPi 3 "I2C1". Die Methoden registerTemperatureSensor sowie registerPressureSensor, werden von der Treiber-Klasse bereitgestellt und sorgen dafür, dass die Daten an den SensorManager korrekt übertragen werden. Der zweite Teil der onCreate-Methode besorgt die Daten des Google Cloud Platform-Projekts (AccessToken und ID) und instanziiert die im nächsten Kapitel beschriebene Funktion zum Datenaustausch mittels Publish/Subscribe-Paradigma.

```
private SensorManager.DynamicSensorCallback mDynamicSensorCallback
        = new SensorManager.DynamicSensorCallback() {
    @Override
    public void onDynamicSensorConnected(Sensor sensor) {
        if (sensor.getType() == Sensor.TYPE_AMBIENT_TEMPERATURE) {
            //Start receiving temperature data.
            mSensorManager.registerListener(mTemperatureListener, sensor,
                    SensorManager.SENSOR_DELAY_NORMAL);
            if (mPubsubPublisher != null) {
                mSensorManager.registerListener(mPubsubPublisher.getTemperatureListener(),
                        sensor, SensorManager.SENSOR_DELAY_NORMAL);
            }
        } else if (sensor.getType() == Sensor.TYPE_PRESSURE) {
            //Start receiving pressure data.
            mSensorManager.registerListener(mPressureListener, sensor,
                    SensorManager.SENSOR_DELAY_NORMAL);
            if (mPubsubPublisher != null) {
                mSensorManager.registerListener(mPubsubPublisher.getPressureListener(), sensor,
                        SensorManager.SENSOR_DELAY_NORMAL);
            }
        }
    }
```

Abbildung 23: Registrierung des Listeners für Temperatur- und Luftdrucksensor[213]

[213] in Anlehnung an: Android Things o. V. (2018).

Der SensorManager besitzt eine abstrakte Klasse, mit einer Callback-Methode, zur Verarbeitung des Konnektivitätsstatus des angeschlossenen Sensors. Die Methode mit der Bezechnung onDynamicSensorConnected, registriert sowohl den Temperatur- als auch dem Luftdrucksensor, jeweils für die Echtzeit-Änderung der Activity und die Publish/Subscribe-Funktion (siehe Abbildung 23). Aufgabe der Listener, in der TemperatureSensorActivity, ist die aktuellen Werte in ein Log zu schreiben und die Daten der momentanen Temperatur und des Luftdrucks anzuzeigen.[214]

6.3.4.3 Publish/Subscribe

Das Publish/Subscribe-Paradigma ist ein im IoT häufig gewähltes Konzept (vgl. hierzu auch Kapitel 3.2.2). Das liegt beispielsweise an der Nachrichtenorientierung des Konzepts. Das Thema (topic) für die Nachrichtenübertragung, im Anwendungsfall der Sensorerfassung von Temperatur- und Luftdruckdaten, ist "raumtemperatur". In der onCreate-Methode der TemperatureSensorActivity wird dieses topic, für die Übertragung, bei der Instanziierung des PubSubPublisher-Objekts übergeben.

Der Konstruktor[215] prüft die in der Anwendung hinterlegten Projekt- und Anwendungsdaten der Google Cloud. Die aus der Anwendung gesendeten Nachrichten werden von der PubSub-Funktion der Cloud Plattform abonniert. Darüber hinaus wird im Konstruktor ein HTTP-Transportobjekt erzeugt.

Ausgeführt wird die Nachrichtenübertragung durch einen nebenläufigen Prozess. Dieser beinhaltet die beiden Methoden run und createMessagePayload. Die createMessagePayload-Methode[216] befüllt das JSON-Objekt, welches als Datenobjekt fungiert. Als Übergabeparameter bekommt diese Methode die zuletzt gelesenen Temperatur- und Luftdruckdaten, von den am Sensor registrierten Listenern. Das JSON-Datenobjekt, mit den Informationen zum Gerät, dem Kanal "pubsub", dem aktuellen Zeitstempel und den Nutzdaten "temperature" und "pressure", wird an die run-Methode zurückgeliefert. Dargestellt in Abbildung 24, zu Beginn des try/catch-Blocks. Sind Daten für das Publishing verfügbar, so wird ein PublishRequest erzeugt. Diesem Request werden die Nachrichten übergeben (setMessages) und abschließend über den Fluent API-Ausdruck (Konkatenation mehrerer Methoden)

---

[214] Vgl. Anhang A3.4.1.
[215] Vgl. Anhang A3.4.2.
[216] Vgl. Anhang A3.4.3.

"projects().topics().publish().execute()" veröffentlicht. Die Häufigkeit ist im vorliegenden Anwendungsfall auf eine Minute begrenzt.

```
@Override
public void run() {
    ConnectivityManager connectivityManager =
            (ConnectivityManager) mContext.getSystemService(Context.CONNECTIVITY_SERVICE);
    assert connectivityManager != null;
    NetworkInfo activeNetwork = connectivityManager.getActiveNetworkInfo();
    if (activeNetwork == null || !activeNetwork.isConnectedOrConnecting()) {
        Log.e(TAG, msg: "no active network");
        return;
    }

    try {
        JSONObject messagePayload = createMessagePayload(mLastTemperature, mLastPressure);
        if (!messagePayload.has( name: "data")) {
            Log.d(TAG, msg: "no sensor measurement to publish");
            return;
        }
        Log.d(TAG, msg: "publishing message: " + messagePayload);
        PubsubMessage m = new PubsubMessage();
        m.setData(Base64.encodeToString(messagePayload.toString().getBytes(),
                Base64.NO_WRAP));
        PublishRequest request = new PublishRequest();
        request.setMessages(Collections.singletonList(m));
        mPubsub.projects().topics().publish(mTopic, request).execute();
    } catch (JSONException | IOException e) {
        Log.e(TAG, msg: "Error publishing message", e);
    } finally {
        mHandler.postDelayed(mPublishRunnable, TimeUnit.MINUTES.toMillis( duration: 1));
    }
}
```

Abbildung 24: Nebenläufiger Thread zum regelmäßigen Publishing der Sensordaten[217]

Nach Veröffentlichung der Daten, werden diese durch den Subscriber der Cloud Pub/Sub-Funktion empfangen und zur Weiterverarbeitung, durch einen automatischen Job, an BigQuery, dem Data Warehouse der Google Cloud Platform, weitergeleitet. Dies geschieht nach einer kurzen Konfiguration automatisch. Allerdings werden die Möglichkeiten eines Data Warehouses erst bei größeren und komplexeren Datenmengen interessant. Unter Berücksichtigung der Zielsetzung dieser Arbeit reicht die fehlerfreie Datenübertragung, über den gesamten Kommunikationsweg, durch das Pub/Sub-Paradigma aus.

[217] in Anlehnung an: Android Things o. V. (2018).

6.3.5 Anwendungsfall Begleitanwendung

6.3.5.1 Grundlagen

Ausgangspunkt der Anwendungsbeispiele Notification und Temperatursteuerung ist, neben der bisher betrachteten Anwendung des Smart Home Prototyps, eine weitere Anwendung auf Basis des Betriebssystems Android. Da auch bei dieser Anwendung die technische Umsetzung der Funktionen im Mittelpunkt steht, werden im Folgenden diese Aspekte genauer beleuchtet.

Grundsätzlich implementiert die Begleitanwendung eine Activity (MainActivity) aus der alle Anwendungsfälle gesteuert werden. Der Abruf der aktuellen Temperaturdaten sowie die Eingabe dieser, wird durch einfache Widgets des Android SDKs umgesetzt.

Für die Nutzung des Firebase Cloud Services wird ein Firebase Projekt benötigt. Dieses kann mit dem bereits genutzten Projekt der Google Cloud Platform verknüpft werden, sodass Datenauswertungen und die integrative Nutzung beider Dienstleistungen ermöglicht wird. Abhängigkeiten in der Gradle-Datei der Begleitanwendung, sind zwei SDKs des Firebase Services für Android - Firebase Messaging und Firebase Database.

6.3.5.2 Anwendungsfall Notification

Nachrichten über Firebase Messaging können auf drei verschiedenen Wegen an die nutzenden Anwendungen übertragen werden. Entweder wird die Nachricht an alle Geräte gesendet, an eine bestimmtes Thema, wie aus dem Publish/Subscribe-Paradigma bekannt, oder an ein spezifisches Gerät. Im vorliegenden Anwendungsfall soll bei einem rapiden Temperaturanstieg z.B. durch ein Feuer, die Meldung vom Smart Home System, über den Firebase Server, an den Nutzer den Begleitanwendung geschickt werden. Die Identifizierung zwischen Smart Home Prototyp und Begleitanwendung erfolgt über einen eindeutigen Token, der zwischen den beiden Parteien ausgetauscht und in der Cloud-Funktion erzeugt und verwaltet wird. Dafür wird die Klasse FirebaseInstanceIdService des Firebase SDKs verwendet. Ein weiterer unabdingbarer Service ist die Basisklasse FirebaseMessagingService. Beide Services werden in der Manifest-Datei der Begleitanwendung eingetragen (Abbildung 25). Bei Adressierung der Anwendung, des Themas oder des Geräts, wird, wie bei

einer Listener-Funktion, automatisch die implementierte Basisfunktion des Firebase-Services ausgeführt.

Beide dargestellten Basisklassen werden, für die an den Anwendungsfall angepasste Nutzung, durch den Entwickler implementiert. Die Klasse MyFirebaseInstanceIdService erweitert die Klasse FirebaseInstanceIdService und überschreibt die Methode onTokenRefresh. Hierüber wird der eindeutige Geräte-Token erstellt, über den Smart Home Prototyp, Firebase und die Begleitanwendung kommunizieren.

```
<service
    android:name=".MyFirebaseMessagingService"
    tools:ignore="ExportedService">
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT"/>
    </intent-filter>
</service>
<service
    android:name="MyFirebaseInstanceIDService"
    tools:ignore="ExportedService">
    <intent-filter>
        <action android:name="com.google.firebase.INSTANCE_ID_EVENT"/>
    </intent-filter>
</service>
```

Abbildung 25: Auszug Manifest-Datei der Begleitanwendung

Die Klasse FirebaseMessagingService wird durch die Klasse MyFirebaseMessagingService erweitert. Aufgabe dieser Klasse ist es, die empfangene Nachricht (RemoteMessage) in einzelne Objekte zu zerlegen (image, title, text und sound) und aufbereitet an das Gerät zu senden (vgl. Abbildung 26). Die Funktion sendNotification[218] nutzt den Android NotificationBuilder, um die aufbereiteten Daten auf dem Smartphone des Anwenders zur Anzeige zu bringen.

Auf der Seite des Smart Home Prototyps, sind keinerlei große Implementierungen vorzunehmen. Lediglich die Abhängigkeit zum FirebaseInstanceIdService ist nötig, damit eine Token-Erstellung, auch für das Smart Home System durchführbar ist. Das Anzeigen von Notifications ist nicht möglich, da das Betriebssystem Android Things diese Funktion aktuell nicht unterstützt (vgl. hierzu auch 5.3.2). Das Senden der Nachricht erfolgt über HTTP an eine fest definierte URL, die sich aus der Basis Fire-

[218] Vgl. Anhang A3.5.1.

base-URL, dem Projektnamen und der Funktion, inklusive des Parameters send, zusammensetzt.

```
@Override
public void onMessageReceived(RemoteMessage message) {

    String image = message.getNotification().getIcon();
    String title = message.getNotification().getTitle();
    String text = message.getNotification().getBody();
    String sound = message.getNotification().getSound();

    int id = 0;
    Object obj = message.getData().get("id");
    if (obj != null) {
        id = Integer.valueOf(obj.toString());
    }

    this.sendNotification(new NotificationData(image, id, title, text, sound));
}
```

Abbildung 26: Firebase-Nachricht empfangen und verarbeiten[219]

6.3.5.3 Anwendungsfall Temperatursteuerung

.Mit dem Anwendungsfall Temperatursteuerung werden zwei technische Aspekte der Entwicklung des Smart Home Prototyps abgedeckt. Das ist die Steuerung der vernetzten Geräte sowohl lokal über den Prototyp als auch dezentral über die Begleitanwendung. Zudem wird diese Funktion über eine Datenbank abgebildet, welche die aktuellen Werte der Räume und Temperaturen hält. Die Realtime Database des Firebase Cloud Services entspricht dabei dem benötigten Anforderungsprofil. Sie ist schemafrei einsetzbar und daher für die prototypische Entwicklung sehr gut geeignet. Darüber hinaus bietet sie über das Firebase Database SDK Listener-Methoden an, welche bei Datenbankänderungen ausgeführt werden können.

Die Identifizierung der beiden Parteien in der Datenbank erfolgt über einen Universally Uniqe Identifier (UUID). Mit dieser UUID, welche sowohl der Prototyp als auch die Anwendung, ab dem Zeitpunkt ihrer Installation besitzen, können die Datensätze adressiert werden. Das zu speichernde Objekt ist in diesem Anwendungsfall einfach aufgebaut und entspricht einem einfachen Java-Objekt. Ziel des Anwendungsfalls ist es, die Temperatur für beliebigen Räume zentral und dezentral einstellen zu können. Daher beinhaltet das Datenobjekt RaumklimaPOJO zwei Attribute - Temperatur und Raum.[220]

---

[219] in Anlehnung an: Carlos (2016).
[220] Vgl. Anhang A3.5.2.

```
public void updateDatabase(String raum, float temperatur) {
    FirebaseDatabase database = FirebaseDatabase.getInstance();
    DatabaseReference myRef = database.getReference(Uuid);

    RaumklimaPOJO data = new RaumklimaPOJO();
    data.setRaum(raum);
    data.setTemperatur(temperatur);

    myRef.setValue(data);
    Log.d(TAG, msg: "Database reference object " + Uuid + "was updated to: " + data);
}
```

Abbildung 27: Einfache Update-Funktion der Realtime Database

Voraussetzung für den Zugriff auf die Funktion der Realtime Database ist wiederum die Prüfung des AccessTokens. Ist diese Prüfung erfolgreich, kann eine Instanz der Realtime Database erzeugt werden und Datensätze, in diesem Fall auf Geräteebene, mit der eindeutigen UUID referenziert werden. Der oben dargestellte Quellcodeausschnitt beschreibt eine einfache Update-Funktion auf die Datenbank. Diese kann einfache Datenobjekte, wie das zuvor beschriebene RaumklimaPOJO, oder JSON-Strukturen verarbeiten.

```
    FirebaseDatabase database = FirebaseDatabase.getInstance();
    DatabaseReference myRef = database.getReference(Uuid);

    myRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            //This method is called every time data in database changes
            RaumklimaPOJO value = dataSnapshot.getValue(RaumklimaPOJO.class);

            //Update data Display
            t.setText(String.valueOf(value != null ? value.getTemperatur() : 0));

            //TODO: Send data to specific Heating Controller for Smart Home via MQTT

            Log.d(TAG, msg: "Value is: " + value);
        }

        @Override
        public void onCancelled(DatabaseError error) {
            Log.w( tag: "TAG", msg: "Failed to read value.", error.toException());
        }
    });
```

Abbildung 28: Implementierung des Datenbank Listeners

Wird durch Interaktion des Nutzers mit dem System, entweder durch Spracheingabe des Smart Home Systems oder durch Eingabe eines Wertes auf der Smartphone-Begleitanwendung, ein Wert geändert, soll dieser auf beiden Systemen sofort verarbeitet werden. Das bedeutet, es muss eine Aktualisierung der Anzeige auf beiden Seiten erfolgen. Auf Seiten des Smart Home Systems zusätzlich noch ein Service-Aufruf z.B. über MQTT, zur Einstellung der gewünschten Temperatur. In beiden

Fällen wird der Listener der Datenbank in der onCreate-Methode der MainActivity registriert (vgl. Abbildung 28). Die Methode onDataChange des Listeners enthält ein Objekt von Typ DataSnapshot. Bei einfachen Datentypen, wie dem RaumklimaPO-JO, findet durch die Methode getValue des DataSnapshot-Objekts, ein automatisches Mapping statt. Nachfolgend können die notwendigen Funktionen zum Abschluss des Anwendungsfalls aufgerufen werden. Im dargestellten Quellcodeausschnitt sind dies, das setzen des Displaytextes sowie der Kommentar zur Ausführung eines Services auf einen vernetzen Heizungscontroller.

7 Ausblick

Der nachfolgend dargestellte Ausblick bezieht sich nicht auf das Themengebiet IoT allgemein, sondern auf die Möglichkeiten der Weiterentwicklung des vorgestellten Smart Home Prototyps.

Auf grober Ebene ist ein Anwender- und Gerätemanagement in einer marktreifen Version des Systems unabdingbar. Diese Funktionalitäten können sehr leicht durch eine Dienstleistung in der Cloud ergänzt werden. Die Google Cloud Platform bietet dafür einen IoT Core-Service an, welcher die verschiedenen IoT-Geräte in einem Netzwerk registrieren und verwalten kann.

Darüber hinaus ist die breitere Nutzung von Datenbanken und Analysemöglichkeiten für ein erfolgreiches Produkt notwendig. Im Rahmen der beispielhaften Umsetzung des Smart Home Prototyps und des Anwendungsfalls Auslesen der Temperatur- und Luftdruckdaten des BMP280 Sensors, ist die Erzeugung einer ausreichenden Menge von Daten zu Analysezwecken nicht zu realisieren. Generell gilt, bei einer Industrieanwendung entstehen deutlich mehr Daten zur Erfassung, als bei einem Smart Home System. Trotzdem können Analysen auch in der Hausautomation anhand von Wetterdaten sinnvoll sein, um z.B. die Heizungssteuerung noch intelligenter zu gestalten. Aus den verwerteten Daten können Prognosen zur Einsparung von Heizkosten und des voraussichtlichen Gesamtverbrauchs an Heizkosten eines Monats/Jahres bestimmt werden. In diesem Fall ist die Betrachtung der möglichen Datenbanken und Analysewerkzeuge, unter Zuhilfenahme der spezifischen Anforderungen, neu zu untersuchen.

Die Integration von Anbietern intelligenter Haushaltssysteme ist ein weiterer Schritt, der zukünftig umgesetzt werden kann. Viele Hersteller bieten ihre eigenen Smart Home Systeme an. Andere hingegen stellen öffentliche oder semi-öffentliche APIs zur Verfügung, mit denen eine Integration von Heizungscontrollern oder Licht- und Kamerasystemen möglich ist.

Auf feingranularer Ebene, sind Ansätze für die Zukunft mit dem Skill Raumklima bereits getroffen worden. Eine neue Funktion stellt jeweils einen Service zur Verfügung. Die Funktion wird in der MainActivity instanziiert und registriert sich so am

System. Die Funktionalität dieses Skills, kann durch die Serviceschnittstelle vollständig aus dem Hauptprozess der Anwendung ausgelagert werden. Der Service verwaltet neben der eigenen Serviceklasse, auch die benötigten Datenobjekte. Erst durch die Integration neuer Funktionen kann die Anwendung den Weg vom Prototyp, hin zu einer praxistauglichen Anwendung machen.

In der Konzeption des Systems wurde festgelegt, dass der Prototyp vollständig ohne GUI lauffähig sein soll. Zur Umsetzung der Funktionen und Anwendungen wurde ein Display lediglich zu Kontrollzwecken eingesetzt. Die Umsetzung einer grafischen Benutzerschnittstelle ist mit dem Android SDK möglich. Für zukünftige Entwicklungen ist die Bereitstellung einer grafischen Oberfläche als Ergänzung zu der natürlichsprachlichen Eingabemöglichkeit denkbar, bedarf aber eines ähnlichen Interaktionskonzepts, wie die vorgestellten Sprachdialoge.

Zuletzt sind Updates ein wichtiger Faktor eines produktiven Systems. Das Betriebssystem Andorid Things unterstützt Updates over-the-air. Das bedeutet, sicherheitsrelevante Lücken des Systems werden durch automatische Updates geschlossen. Dafür muss das System lediglich einen Netzwerkzugang besitzen. In der Entwicklervorschau gibt es noch keine Hinweise dazu, wie Anwendungen eines Systems Updates ausrollen können. Sofern die Funktionen ausgelagert sind, ist ein Update der Anwendung auf dem Gerät nicht notwendig. Sollte jedoch bspw. ein neues Sprachmodell von Pocketsphinx installiert werden, ist der Zugriff auf die Anwendung erforderlich.

8 Zusammenfassung und Fazit

Zusammenfassend wurden im Umfang dieser Arbeit viele unterschiedliche Gesichtspunkte des Themenkomplexes Internet of Things betrachtet. Die Einschätzung, dass sich unter dem Schlagwort IoT sehr viele verschiedene Entwicklungen der Informationstechnologie sammeln, wurde bestätigt.

Die Entstehungsgeschichte des Internets hat aufgezeigt, wie rasant die Entwicklung in den letzten Jahren vorangeschritten ist. Diese Entwicklung ermöglicht neue Anwendungsmöglichkeiten und Chancen für das alltägliche Leben der Menschen. Durch Gesundheitsüberwachung mittels Wearables sowie das Wohnen und Arbeiten in intelligenten Gebäuden und Städten, ausgestattet mit Sensoren, kann eine nachhaltige Veränderung der Situation herbeigeführt werden. Aber auch in der Industrie sind die Effekte von verbesserter Produktionsleistung, hin zu neuen Geschäftsmodellen anzuführen. Smart Home Systeme können den Alltag erleichtern und bei der Einsparung von Energie und Kosten im Haushalt unterstützen. Bei all diesen Entwicklungen ist immer das Thema Datenschutz und Datensicherheit zu berücksichtigen. Veraltete Gesetzesregelungen, im Kontrast zur technologischen Weiterentwicklung, sind gefährliche Zusammensetzungen in diesem Kontext. Um hier eine allgemeingültige Regelung zu finden müssen sowohl Gesetzesgeber als auch Unternehmen und Anwender sich bewusst sein, welche Daten sie preisgeben möchten und welche Daten vom System überhaupt gespeichert werden.

Die Technologien des Internet of Things basieren zu großen Teilen auf bereits bekannten Paradigmen. Allerdings zwingen die Eigenschaften der IoT-Hardware, z.B. einfache Mikrocontroller, diese dazu, effizienter zu arbeiten oder neue Wege zu beschreiten, um die Datenübertragungsraten zu senken oder den Energieverbrauch der Kommunikation zu verringern. Genannte Beispiele sind BLE oder das nachrichtenorientierte Protokoll MQTT.

Das Thema Datenmengen ist im Kontext dieser Arbeit nur am Rande untersucht worden. Die Schlagwörter NoSQL und Big Data sind in diesem Zusammenhang zu nennen. Eine skalierbare Infrastruktur sowie die Möglichkeit zur Speicherung und Analyse heterogener Datenbestände ist die Herausforderung. Wie bereits im Ausblick beschrieben, ist die Analyse solch großer Datenbestände bei der Umsetzung

eines einzelnen Prototyps, ohne Integration weiterer Systeme, fachlich nicht sinnvoll umzusetzen.

Des Weiteren wurde die Funktionsweise sowie die Chancen und Herausforderungen von natürlichsprachlichen Dialogsystemen aufgezeigt. Diese sind für das IoT unerlässlich, da neue Benutzerschnittstellen geschaffen werden müssen, die nicht von der visuellen Repräsentation abhängen. Die Fortschritte der letzten Jahre durch DNNs und Machine Learning-Algorithmen, haben das Potenzial dieser Mensch-Maschine Schnittstelle verdeutlicht. Angebote, wie die in der Anwendung genutzte Cloud-Funktionalität Speech-To-Text oder Pocketsphinx, zeigen, dass diese Systeme, auch für die Breite in der Entwicklung, marktreife besitzen.

Besonders die beiden letztgenannten Werkzeuge haben einen großen Einfluss auf die Konzeption und Umsetzung des Smart Home Prototyps gehabt. Auf der einen Seite zeigt die vollständige Offline Wakeword-Erkennung, dass die Angst vor einer permanenten "Wanze" im Wohnzimmer bei Smart Home Systemen nicht begründet sein muss, da Alternativen zu den großen Online-Systemen (z.B. Amazon Alexa und Google Home) geschaffen werden können. Auf der anderen Seite ist die genaue Spracherkennung, zur Verbesserung der Benutzererfahrung, durch Einsatz des Speech-To-Text Services sehr leicht möglich, ohne dass ein komplexes Training der Sprachmodelle erforderlich ist. Dagegen setzt zusammenzufassend, das Handling der vielen asynchronen Prozesse der Werkzeuge, speziell auf einer vergleichsweise schwachen Hardware, wie dem RaspberryPi 3 Model B, eine genaue Steuerung der nebenläufigen Anwendungen voraus.

Fazit zum Thema Technologien des IoT, unter Berücksichtigung der Umsetzung des Smart Home Prototyps, ist, dass es nicht genau eine Platform oder ein System gibt, welches alle Anforderungen einer Anwendung abdeckt. Auch, wenn sich de facto-Standards für das IoT am Markt etablieren, ist das Prototyping ein sinnvoller Weg die verschiedenen Technologien auf Nutzbarkeit in der eigenen Anwendung zu testen.

Die Entscheidung für Cloud Services, welche allesamt Google-Funktionalitäten integrieren, lässt sich als sinnvoll bewerten. Die Integration der SDKs ist problemlos und die Funktionen sind bereits marktreif. Dadurch ist ein stabiles Fundament für die

Erweiterung der Funktionen des Smart Home Systems geschaffen. Eine Einbindung weiterer Services sowie die Skalierung ist stufenlos möglich.

Die Bewertung des Betriebssystems Android Things fällt hingegen komplexer aus. Vorteile sind die erprobten Entwicklungsstandards und SDKs sowie die Erweiterung dieser Funktionalitäten durch Komponenten des IoT. Allerdings hat die Entwicklervorschau auch Probleme aufgezeigt. So ist es häufiger zu Inkompatibilitäten zwischen der Version der Entwicklervorschau und den aktuellsten Treibern gekommen. Die fehlerfreie Verwendung ist somit nicht über den gesamten Entwicklungszyklus gewährleistet gewesen. Darüber hinaus hat sich im Laufe der verschiedenen Entwicklerversionen die API verändert, was zu Fehlern beim Kompilieren und Bauen der Anwendung geführt hat. Diese Änderungen sind aber hauptsächlich auf den frühen Entwicklungsstatus der ersten Versionen zurückzuführen. Zudem werden nach aktuellem Stand nur einzelne Sensoren mit Treibern unterstützt. Die Anbindung des Temperatur- und Luftfeuchtigkeitssensors DHT22, ist aufgrund einer nicht unterstützten Taktrate des Sensor-Bauteils nicht möglich. Die Anbindung des Temperatur- und Luftdrucksensors BMP280 hat grundsätzlich fehlerfrei funktioniert.

In der Gesamtbetrachtung wurde die Zielsetzung, der Erstellung eines lauffähigen Smart Home Prototyps, mit Einbindung verschiedener Technologien, erreicht. Die technische Funktionalität des Dialogsystems, bestehend aus der Pocketsphinx Wakeword-Erkennung, dem Speech-To-Text ASR-System und dem Dialogflow Dialogmanager, ist gegeben. Unter Betrachtung der funktionalen und nicht-funktionalen Anforderungen an ein Dialogsystem, sind generell keine Hemmnisse der Performanz oder fehlender Genauigkeit, durch Kombination der Werkzeuge, festzustellen. Auch die Anbindung eines Sensors, der Begleitanwendung, zur dezentralen Steuerung des Systems, sowie Echtzeit-Datenbankzugriffe wurden realisiert.

Abschließend soll die Frage aus dem ersten Absatz der Zielsetzung und Motivation aufgegriffen werden. IoT - Was ist Realität und was ist Hype? Diese Frage ist nicht vollumfänglich zu beantworten, da die Themen, welche sich um dieses Thema ranken, an einigen Stellen noch unübersichtlich sind und sich weiter entwickeln müssen. Aber, IoT ist bereits im Alltag angekommen und hat das Potenzial große Auswirkungen auf die Zukunft unserer Gesellschaft, wie wir arbeiten und wie wir leben, zu haben. Technologische und wirtschaftliche Treiber, streben nach immer neuen

Innovationen. Aufgabe des Gesetzgebers ist es, diese in einem sinnvollen Maß kontrollierbar zu halten. Die Entwicklung des Betriebssystems Android Things, auf Basis des marktführenden mobilen Betriebssystems Android, ist zudem ein Indiz dafür, dass IoT Realität ist. Mit diesem Schritt wird unterstrichen, dass die Entwicklung von IoT-Anwendungen nicht nur Spezialisten der Hardware-Programmierung ermöglicht werden soll, sondern eine möglichst große Gruppe von Software-Spezialisten an das Thema IoT-Entwicklung herangeführt wird.

Die Umsetzung des Smart Home Systems auf Basis des kostengünstigen und mit einfacher Hardware ausgestatteten Raspberry Pi, unter Nutzung von Android Things, zeigt, dass dies Realität ist.

9 Literaturverzeichnis

[Abdmeziem, Riad; Tandjaoui, Djamel (2014)]
Internet of Things: Concept, Building blocks, Applications and Challanges, https://arxiv.org/pdf/1401.6877.pdf, abgerufen am 05.05.2018.

[Adafruit o. V. (2015)]
Adafruit BMP280 Barometic Pressure + Temperature Sensor Breakout, https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf, abgerufen am 09.05.2018.

[Adelfinger, Volker P.; Hänisch, Till (2015)]
Grundlagen: Das Internet der Dinge, in: Adelfinger, Volker P.; Hänisch, Till (Hrsg.): Internet der Dinge - Technik, Trends und Geschäftsmodelle, Wiesbaden: Springer Gabler, 2015, S. 9 - 75.

[Alby, Tom (2008)]
Web 2.0 - Konzepte, Anwendungen, Technologien, 3. Aufl., München: Carl Hanser Verlag, 2008.

[Amazon o. V. (o. J. a)]
AWS IoT Device SDK, https://aws.amazon.com/de/iot/sdk/, abgerufen am 01.04.2018.

[Amazon o. V. (o. J. b)]
Design – Alexa Voice Service, https://developer.amazon.com/de/alexa-voice-service/design, abgerufen am 25.04.2018.

[Android Developers o. V. (o. J.)]
Android Things - About the platform, https://developer.android.com/things/, abgerufen am 07.05.2018.

[Android Developers o. V. (2018a)]
Android Things - Android Things Behaviour Changes, https://developer.android.com/things/versions/things-1.0-changes, abgerufen am 07.05.2018.

[Android Developers o. V. (2018b)]
Android Things - Overview, https://developer.android.com/things/get-started/, abgerufen am 07.05.2018.

[Android Developers o. V. (2018c)]
Android Things - Supported platforms, https://developer.android.com/things/hardware/, abgerufen am 07.05.2018.

[Android Things o. V. (2018)]
Android Things - Supported platforms, https://developer.android.com/things/hardware/, abgerufen am 07.05.2018.

[Bartolo, Anthony (2017)]
Dank IoT und Konnektivität - Smart Cities: Vom Traum zur Realität, https://www.computerwoche.de/a/smart-cities-vom-traum-zur-realitaet,3330906, abgerufen am 03.04.2018.

[Bell, Charles (2016)]
MySQL for the Internet of Things - Data Management for sensors and connected devices, 1. Aufl., New York: Springer Science+Business Media, 2016.

[Bendel, Oliver (2018)]
Smart Home, https https://wirtschaftslexikon.gabler.de/definition/smart-home-54137/version-277191, abgerufen am 05.04.2018.

[Berners-Lee, Tim (1998)]
The World Wide Web: A very short personal history, https://www.w3.org/People/Berners-Lee/ShortHistory.html, abgerufen am 18.03.2018.

[Berners-Lee, Tim (2006)]
Design Issues - Linked Data, https://www.w3.org/DesignIssues/LinkedData.html, abgerufen am 28.03.2018.

[Beuth, Patrick; et al. (2018)]
DSGVO: Das sollten Sie zur Datenschutz-Grundverordnung der EU wissen, http://www.spiegel.de/netzwelt/web/dsgvo-das-sollten-sie-zur-datenschutz-grundverordnung-der-eu-wissen-a-1205985.html, abgerufen am 05.05.2018.

[Bial, Dominik; Rock, Daniel (2018)]
IoT in der Cloud - Erkenntnisse und Erfahrungen eines Plattformvergleichs, https://www.informatik-aktuell.de/betrieb/virtualisierung/iot-in-der-cloud-erkenntnisse-und-erfahrungen-eines-plattformvergleichs.html, abgerufen am 04.05.2018.

[Bormann, Carsten (o. J.)]
CoAP - RFC 7252 Constrained Application Protocol, http://coap.technology/, abgerufen am 04.05.2018.

[Branscombe, Mary (2017)]
Beyond the Switchboard: The Current State of the Art in Speech Recognition, https://thenewstack.io/speech-recognition-getting-smarterstate-art-speech-recognition/, abgerufen am 14.04.2018.

[Brauchli, Andreas; Li, Depeng (2016)]
A Solution-Based Analysis of Attack Vectors on Smart Home Systems, in: Hu, Fei (Hrsg.): Security and Privacy in Internet of Things (IoTs) - Models, Algorithms, and Implementations: Taylor & Francis, 2016, S. 91 - 106.

[Cabe, Benjamin (2018)]
Key Trends from the IoT Developer Survey 2018, https://blog.benjamin-cabe.com/2018/04/17/key-trends-iot-developer-survey-2018, abgerufen am 05.05.2018.

[Carlos, Antonio (2016)]
GitHub - Firebase Cloud Messaging tutorial, https://github.com/ton1n8o/FCMTutorial, abgerufen am 17.05.2018.

[Carstensen, Kai-Uwe; Jekat, Susanne; Klabunde, Ralf (2010)]
Computerlinguistik - Was ist das?, in: Carstensen, Kai-Uwe; et al. (Hrsg.): Computerlinguistik und Sprachtechnologie - Eine Einführung, Heidelberg: Spektrum Akademischer Verlag, 3. Auflage, 2010, S. 1 - 26.

[CERN (2013)]
The birth of the Web, http://cds.cern.ch/record/1998446, abgerufen am 18.03.2018.

[Chellapan, V.; Sivalingam, K. M. (2016)]
Chapter 10: Security and Privacy in the Internet of Things, in: Buyya, Rajkumar; Dastjerdi, Amir V. (Hrsg.): Internet of Things - Principles and Paradigms, Cambridge: Elsevier, 2016, S. 183 - 200.

[Cisco (o. J.)]
Internet of Everything, http://ioeassessment.cisco.com/de, abgerufen am 29.03.2018.

[CMUSphinx o.V. (o. J. a)]
Basic concepts of speech recognition, https://cmusphinx.github.io/wiki/tutorialconcepts/, abgerufen am 25.04.2018.

[CMUSphinx o.V. (o. J. b)]
Training an acoustic model for CMUSphinx, https://cmusphinx.github.io/wiki/tutorialam/, abgerufen am 25.04.2018.

[CMUSphinx o.V. (o. J. c)]
Overview of the CMUSphinx toolkit, https://cmusphinx.github.io/wiki/tutorialoverview/, abgerufen am 07.05.2018.

[CMUSphinx o.V. (o. J. d)]
PocketSphinx on Android, https://cmusphinx.github.io/wiki/tutorialandroid/, abgerufen am 07.05.2018.

[CMUSphinx o.V. (o. J. e)]
GitHub - cmusphinx/pocketsphinx-android-demo, https://cmusphinx.github.io/wiki/tutorialandroid/, abgerufen am 07.05.2018.

[CXP Group (2017)]
Digital industrial transformation with the Internet of Things - Trend report, https://www.pac-online.com/download/23125/181580/?ct=0, abgerufen am 01.04.2018.

[DB-Engines o. V. (2018)]
DB-Engines Ranking, https://db-engines.com/de/ranking, abgerufen am 04.05.2018.

[Deloitte (2016)]
Industrielles Internet der Dinge und die Rolle von Telekommunikationsunternehmen - Hype oder vernetzte Revolution?, https://www2.deloitte.com/content/dam/Deloitte/de/Documents/technology-media-telecommunicatons/Deloitte_TMT_Industrielles%20Internet%20der%20Dinge.pdf, abgerufen am 01.04.2018.

[Destatis (2016)]
81% der Internetnutzer gehen per Handy oder Smartphone ins Internet, https://www.destatis.de/DE/PresseService/Presse/Pressemitteilungen/2016/12/PD16_430_63931.html, abgerufen am 18.03.2018.

[Dialogflow o. V. (2018a)]
Basics, https://dialogflow.com/docs/getting-started/basics, abgerufen am 25.04.2018.

[Dialogflow o. V. (2018b)]
Integrations, https https://dialogflow.com/docs/integrations/, abgerufen am 09.05.2018.

[Dialogflow o. V. (2018c)]
Agent Design, https://dialogflow.com/docs/best-practices/agent-design, abgerufen am 09.05.2018.

[Dialogflow o. V. (2018d)]
Slot Filling, https://dialogflow.com/docs/how-tos/slot-filling, abgerufen am 09.05.2018.

[DSGVO o. V. (o. J.)]
Datenschutz durch Technikgestaltung und durch datenschutzfreundliche Voreinstellungen, https://dsgvo-gesetz.de/art-25-dsgvo/, abgerufen am 05.05.2018.

[Ebert, Christian; Ebert, Cornelia (2010)]
Methoden, in: Carstensen, Kai-Uwe; et al. (Hrsg.): Computerlinguistik und Sprachtechnologie - Eine Einführung, Heidelberg: Spektrum Akademischer Verlag, 3. Auflage, 2010, S. 169 - 479.

[Eclipse Foundation o. V. (2018a)]
Open source stack for IoT Gateways & Smart Devices, https://iot.eclipse.org/gateways/, abgerufen am 28.04.2018.

[Eclipse Foundation o. V. (2018b)]
Open source stack for IoT Cloud Platforms, https://iot.eclipse.org/cloud/, abgerufen am 28.04.2018.

[Edlich, Stefan; et al. (2011)]
NoSQL - Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken, 2. Aufl., München: Carl Hanser Verlag, 2011.

[Farzad, Khodadadi; et. al (2016)]
Chapter 1: Internet of Things: An Overview, in: Buyya, Rajkumar; Dastjerdi, Amir V. (Hrsg.): Internet of Things - Principles and Paradigms, Cambridge: Elsevier, 2016, S. 5 - 28.

[Fiducia & GAD IT AG o. V. (2018)]
Was ist eigentlich die Fiducia & GAD IT AG?, https://www.fiduciagad.de/ueber-uns.html, abgerufen am 17.05.2018.

[Firebase o. V. (2018a)]
Firebase Cloud Messaging, https://firebase.google.com/docs/cloud-messaging/, abgerufen am 10.05.2018.

[Firebase o. V. (2018b)]
Firebase Realtime Database, https://firebase.google.com/docs/database/, abgerufen am 10.05.2018.

[Fraunhofer ESK (2016)]
Industrial Internet of Things: Referenzarchitektur für die Kommunikation, https://www.esk.fraunhofer.de/content/dam/esk/dokumente/Whitepaper_IoT_dt_April16.pdf, abgerufen am 04.04.2018.

[Fraunhofer Fokus (2016)]
Public IoT - Das Internet der Dinge im öffentlichen Raum, https://cdn0.scrvt.com/fokus/36c5e4909a46af02/982714594f78/WP_Public_Internet_of_Things_web.pdf, abgerufen am 03.04.2018.

[Gartner (2017a)]
Gartner Hype Cycle for Ermerging Technologies, 2017, https://blogs.gartner.com/smarterwithgartner/files/2017/08/Emerging-Technology-Hype-Cycle-for-2017_Infographic_R6A.jpg, abgerufen am 28.03.2018.

[Gartner (2017b)]
Leading the IoT - Gartner Insights on How to Lead in a Connected World, https://www.gartner.com/imagesrv/books/iot/iotEbook_digital.pdf, abgerufen am 30.03.2018.

[Gartner (2017c)]
Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016, https://www.gartner.com/newsroom/id/3598917, abgerufen am 01.04.2018.

[Gartner o. V. (o. J.)]
Big Data, https://www.gartner.com/it-glossary/big-data/, abgerufen am 04.05.2018.

[Gezetic, Elma (2017)]
Comparison Between Cloud-based and Offline Speech Recognition Systems, https://mediatum.ub.tum.de/doc/1399984/1399984.pdf, abgerufen am 15.04.2018.

[Google (o. J.)]
CLOUD IOT CORE, https://cloud.google.com/iot-core/, abgerufen am 01.04.2018.

[Google o. V. (2018)]
Build Your First App with Dialogflow, https://developers.google.com/actions/dialogflow/first-app, abgerufen am 25.04.2018.

[Google Cloud o. V. (2018)]
Was ist Google Cloud Pub/Sub?, https://cloud.google.com/pubsub/docs/overview, abgerufen am 10.05.2018.

[Google Cloud o. V. (o. J. a)]
Cloud Speech-To-Text, https://cloud.google.com/speech-to-text/, abgerufen am 07.05.2018.

[Google Cloud o. V. (o. J. b)]
Google BigQuery, https://cloud.google.com/bigquery/, abgerufen am 10.05.2018.

[Howard, Philip N. (2016)]
Finale Vernetzung - Wie das Internet der Dinge unser Leben verändern wird, 1. Aufl., Köln: Bastei Lübbe, 2016.

[Höller, Jan; et al. (2014)]
From Machine-to-Machine to the Internet of Things - Introduction to a New Age of Intelligence, 1. Aufl., Oxford: Elsevier, 2014.

[IT Wissen o.V. (2013)]
ASR (automatic speech recognition), https://www.itwissen.info/ASR-automatic-speech-recognition-Automatische-Spracherkennung.html, abgerufen am 14.04.2018.

[IT Wissen o.V. (2016)]
Smart Wearables, https://www.itwissen.info/Smart-Wearables-smart-wearables.html, abgerufen am 05.04.2018.

[i-Scoop o. V. (o. J.)]
Consumer Internet of Things (CIoT) - what is it and how does it evolve?, https://www.i-scoop.eu/internet-of-things-guide/what-is-consumer-internet-of-things-ciot/, abgerufen am 05.04.2018.

[Karasiewicz, Christian (2013)]
Why HTTP is not enough for the Internet of Things, https://www.ibm.com/developerworks/community/blogs/mobileblog/entry/why_http_is_not_enough_for_the_internet_of_things?lang=en, abgerufen am 04.05.2018.

[Kaufmann, Timothy (2015)]
Geschäftsmodelle in Industrie 4.0 und dem Internet der Dinge - Der Weg vom Anspruch in die Wirklichkeit, 1. Aufl., Wiesbaden: Springer Fachmedien, 2015.

[Këpuska, Veton; Bohuta, Gamal (2017)]
Comparing Speech Recognition Systems (Microsoft API, Google API And CMU Sphinx), in: International Journal of Engineering Research and Application, ISSN 2248-9622, S. 20 - 24.

[Kleuker, Stephan (2018)]
Grundkurs Software Engineering mit UML - Der pragmatische Weg zu erfolgreichen Softwareprojekten, 4. Aufl., Wiesbaden: Springer Fachmedien, 2018.

[Kobylinska, Anna; Martins, Felipe Pereira (2017)]
Betriebssysteme für das Internet der Dinge, https://www.com-magazin.de/praxis/internet-dinge/betriebssysteme-internet-dinge-1222988.html, abgerufen am 04.05.2018.

[Kranz, Maciej (2017)]
Building The Internet Of Things, 1. Aufl., Hoboken, New Jersey: John Wiley & Sons, 2017.

[Khandelwal, Swali (2017)]
Bluetooth Hack Affects 20 Million Amazon Echo and Google Home Devices, https://thehackernews.com/2017/11/amazon-alexa-hacking-bluetooth.html, abgerufen am 05.05.2018.

[Lea, Perry (2018)]
Internet of Things for Architects, 1. Aufl., Birmingham: Packt Publishing, 2018.

[Lei, Xin; et al. (2013)]
Accurate and Compact Large Vocabulary Speech Recognition on Mobile Devices, https://static.googleusercontent.com/media/research.google.com/de//pubs/archive/41176.pdf, abgerufen am 25.04.2018.

[Leiner, Barry M.; et al. (1997)]
Brief History of the Internet, https://www.internetsociety.org/internet/history-internet/brief-history-internet/, abgerufen am 18.03.2018.

[Litzel, Nico (2016)]
Was ist Natural Language Processing?, https://www.bigdata-insider.de/was-ist-natural-language-processing-a-590102/, abgerufen am 14.04.2018.

[Ludewig, Jochen; Lichter, Horst (2013)]
Software Engineering - Grundlagen, Menschen, Prozesse, Techniken, 3. Aufl., Heidelberg: dpunkt.verlag, 2013.

[Marmonti, Stefano (2016)]
Relationale Datenbanken sind ungeeignet, https://www.bigdata-insider.de/relationale-datenbanken-sind-ungeeignet-a-523398/, abgerufen am 04.05.2018.

[Mattern, Friedemann (2005)]
Teil A: Visionen - Die technische Basis für das Internet der Dinge, in: Fleisch, Elgar; Mattern, Friedemann (Hrsg.): Das Internet der Dinge - Ubiquitous Computing und RFID in der Praxis, Berlin: Springer-Verlag, 2005, S. 38 - 66.

[Mattern, Friedemann; Floerkemeier, Christian (o. J.)]
From the Internet of Computers to the Internet of Things, http://vs.inf.ethz.ch/publ/papers/Internet-of-things.pdf, abgerufen am 30.03.2018.

[McDonley, Alan (2016)]
Raspberry Pi 3 Model B Roadtest - Pocket Sphinx Speech Recognition Performance Comparison, https://www.element14.com/community/roadTestReviews/2166/l/roadtest-review-a-raspberry-pi-3-model-b-review, abgerufen am 07.05.2018.

[McGraw, Ian; et al. (2016)]
Personalized Speech Recognition on Mobile Devices, https://arxiv.org/pdf/1603.03185.pdf, abgerufen am 15.04.2018.

[Meier, Andreas (2018)]
Werkzeuge der digitalen Wirtschaft: Big Data, NoSQL & Co. - Eine Einführung in relationale und nicht-relationale Datenbanken, 1. Aufl., Wiesbaden: Springer Fachmedien, 2018.

[Militaru, Diana; Gavat, Inge (2014)]
A historically perspective of speaker-independent speech recognition in Romanian language, https://www.researchgate.net/publication/299594444_A_historically_perspective_of_speaker-independent_speech_recognition_in_Romanian_language, abgerufen am 25.04.2018.

[Mukhopadhyay, S. C.; Suryadevara, N. K. (2014)]
Internet of Things: Challenges and Opportunities, in: Mukhopadhyay, Subhas C. (Hrsg.): Internet of Things - Challenges and Opportunities, Cham: Springer Science+Business Media, 2014, S. 1 - 18.

[Müller, Stefan (2016)]
Internet of Things (IoT) - Ein Wegweiser durch das Internet der Dinge, 1. Aufl., Norderstedt: BoD - Books on Demand, 2016.

[Nicholas, Stephen (2012)]
Power Profiling: HTTPS Long Polling vs. MQTT with SSL, on Andorid, http://stephendnicholas.com/posts/power-profiling-mqtt-vs-https, abgerufen am 04.05.2018.

[Nowosak, Samira (2018)]
Was die EU-DSGVO für Smart Home bedeutet, https://www.homeandsmart.de/dsgvo-datenschutzgrundverordnung-smart-home, abgerufen am 05.05.2018.

[NXP Semiconductors o. V. (2014)]
UM10204 - I2C-bus specification and user manual, https://www.nxp.com/docs/en/user-guide/UM10204.pdf, abgerufen am 02.05.2018.

[O'Reilly, Tim (2005)]
What Is Web 2.0, http://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html, abgerufen am 26.03.2018.

[Pearl, Cathy (2016)]
Designing Voice User Interfaces - Principles of Conversational Experiences, 1. Aufl., Gravenstein: O'Reilly, 2016.

[Pettey, Christy (2016)]
Building a Blueprint for the Internet of Things, https://www.gartner.com/smarterwithgartner/build-a-blueprint-for-the-internet-of-things/, abgerufen am 04.05.2018.

[Pütter, Christiane (2018)]
Die 10 wichtigsten IT-Trends bis 2018, https://www.cio.de/a/die-10-wichtigsten-it-trends-bis-2018,3247944, abgerufen am 25.04.2018.

[Protalinski, Emil (2017)]
Google's speech recognition technology now has a 4.9% word error rate, https://venturebeat.com/2017/05/17/googles-speech-recognition-technology-now-has-a-4-9-word-error-rate/, abgerufen am 25.04.2018.

[Raspberry Pi Foundation o. V. (o. J.)]
Raspberry Pi 3 Model B+, https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf, abgerufen am 09.05.2018.

[Ray, Brian (2015)]
3 Reasons Why Ipv6 Is Important For The Internet Of Things, https://www.link-labs.com/blog/why-ipv6-is-important-for-internet-of-things, abgerufen am 30.03.2018.

[Retresco o.V. (2017)]
Speech Recognition APIs im Überblick, https://www.retresco.de/speech-recognition-ueberblick/, abgerufen am 25.04.2018.

[Rochow, Marc (2012)]
Web 3.0 - Das semantische Web, http://www.gironimo.org/webentwicklung/web-3-0-das-semantische-web.html, abgerufen am 28.03.2018.

[Roth, Stephan (2016)]
Zuverlässige Datenkommunikation im (Industrial) Internet of Things mit DDS, https://www.informatik-aktuell.de/betrieb/netzwerke/zuverlaessige-datenkommunikation-im-industrial-internet-of-things-mit-dds.html, abgerufen am 04.05.2018.

[Roth, Wolf-Dieter (2006)]
Tim Berners-Lee zum Hype des "neuen Web", https://www.heise.de/tp/features/Web-2-0-ist-nutzloses-Blabla-das-niemand-erklaeren-kann-3407761.html, abgerufen am 26.03.2018.

[Schnellbacher, Mascha (2016)]
Internet of Things, Smart Cities und Schutz der Privatsphäre, https://entwickler.de/online/iot/internet-things-smart-cities-schutz-privatsphaere-214395.html, abgerufen am 03.04.2018.

[Schnelle-Walka, Dirk; et al. (2016)]
NLU vs. Dialog Management: To Whom am I Speaking, https://www.inf.uni-hamburg.de/en/inst/ab/lt/publications/2016-schnelleetal-workshop-smartobjects.pdf, abgerufen am 25.04.2018.

[Scholing, Sascha (2015)]
Internet of Things - Relationale und NoSQL-Datenbanken wachsen zusammen, https://www.bigdata-insider.de/relationale-und-nosql-datenbanken-wachsen-zusammen-a-498946/, abgerufen am 04.05.2018.

[Schricks, Arnulf D. (2017)]
Die Arbeitswelt 4.0 kompetent gestalten, in: Schrick, Arnulf D.; Drenth, Randy; Schneider, Roland (Hrsg.): Strategie für Industrie 4.0 - Praxiswissen für Mensch und Organisation in der digitalen Transformation, Wiesbaden: Springer Fachmedien, 2017, S. 1 - 34.

[Serozhenko, Marina (2017)]
MQTT vs. HTTP: which one is the best for IoT, https://medium.com/mqtt-buddy/mqtt-vs-http-which-one-is-the-best-for-iot-c868169b3105, abgerufen am 04.05.2018.

[Serpanos, Dimitrios; Wolf, Marilyn (2018)]
Internet-of-Things (IoT) Systems - Architectures, Algorithms, Methodologies, 1. Aufl., Cham: Springer International Publishing, 2018.

[Smith, Dave (2018)]
Android Things Release Candidate, https://android-developers.googleblog.com/2018/04/android-things-release-candidate.html, abgerufen am 07.05.2018.

[Stankovic, Milan; Jovanovic, Jelena (2010)]
Section 1: Tagging and Semantics, in: Devedzic, Vladan; Gasevic, Dragan (Hrsg.): Web 2.0 & Semantic Web, New York: Springer Science+Business Media, 2010, S. 2 - 54.

[Statista (2018)]
Anzahl der Internetnutzer weltweit, https://de.statista.com/statistik/daten/studie/805920/umfrage/anzahl-der-internetnutzer-weltweit/, abgerufen am 18.03.2018.

[Sullivan, Dan; Sullivan, James (2016)]
Übersicht und Vergleich: Datenbankoptionen für das Internet of Things (IoT), https://www.searchenterprisesoftware.de/lernprogramm/Uebersicht-und-Vergleich-Datenbankoptionen-fuer-das-Internet-of-Things-IoT, abgerufen am 04.05.2018.

[Sun, Charles (2016)]
No IoT without IPv6, https://www.computerworld.com/article/3071625/internet-of-things/no-iot-without-ipv6.html, abgerufen am 30.03.2018.

[Thar, Evelyn (2015)]
"Ich habe Sie leider nicht verstanden" - Linguistische Optimierungsprinzipien für die mündliche Mensch-Maschine-Interaktion, 1. Aufl., Bern: Peter Lang, 2015.

[Thumar, Brijesh (2017)]
GitHub - Thumar/SpeechAPI, https://github.com/Thumar/SpeechAPI, abgerufen am 07.05.2018.

[Vasseur, Jean-Philippe; Dunkels, Adam (2010)]
Interconnecting Smart Objects with IP, 1. Aufl., Burlington: Elsevier, 2010.

[Wendel, Mariella (2017)]
Smart Home Systeme im Leistungsvergleich, https://www.homeandsmart.de/smart-home-systeme-vergleich, abgerufen am 05.04.2018.

[XMPP o. V. (o. J.)]
About XMPP, https://xmpp.org/about/, abgerufen am 04.05.2018.

[Zamann, Safdar; Slany, Wolfgang (2014)]
Smartphone-Based Online and Offline Speech Recognition System for ROS-Based Robots, in: Tiwari, Namita; Padhye, Sahadeo; He, Debiao (Hrsg.): Information technology and Control, ISSN 2335–884X (online), S. 371 - 380.

[Zhang, Guoming; et al. (2017)]
DolphinAttack: Inaudible Voice Commands, https://acmccs.github.io/papers/p103-zhangAemb.pdf, abgerufen am 05.05.2018.

IV Anhang

A1 Pocketsphinx - Wörterbuch (Auszug)

Zahlenwort	Graphem (Graphendarstellung eines Phonems)
One	W AH N
One (2)	HH W AH N
Two	T UW
Three	TH R IY
Four	F AO R
Five	F AY V
Six	S IH K S
Seven	S EH V AH N
Eight	EY T
Nine	N AY N
Ten	T EH N

Abbildung 29: Pocketsphinx Phonem-zu-Wort Paare[221]

[221] in Anlehnung an: CMUSphinx o. V. (o. J. d).

A2 Screenshots

A2.1 Dialogflow

A2.1.1 Konsole

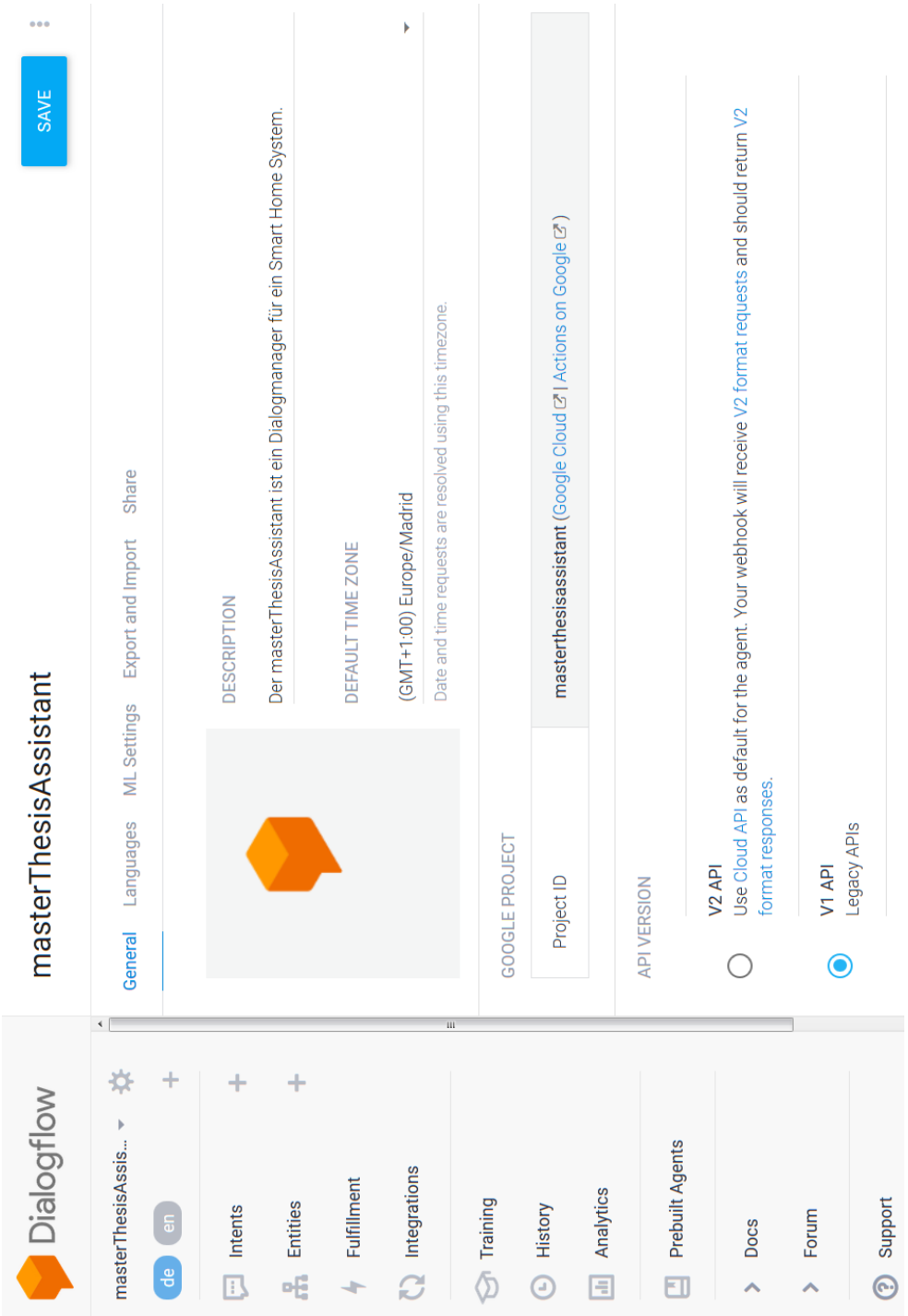

Abbildung 30: Konsole zur Verwaltung und Konfiguration des Dialogflow-Projekts

A2.1.2 Training des Dialogmanagers

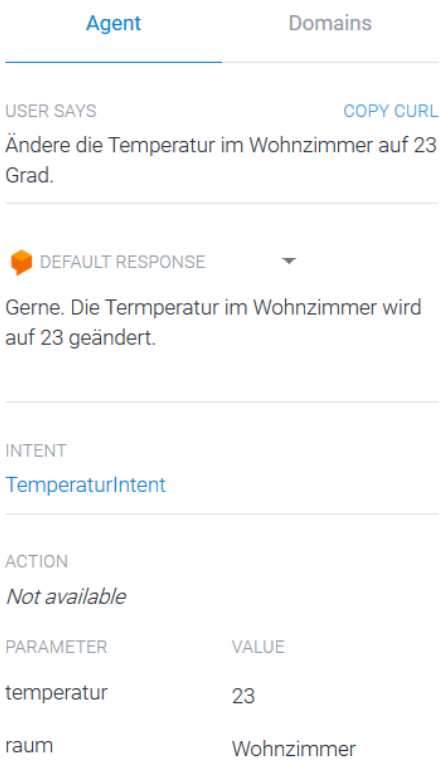

Abbildung 31: Trainingsmodell des Dialogmanagers

A2.1.3 Modellierung Antworten des Dialogmanagers

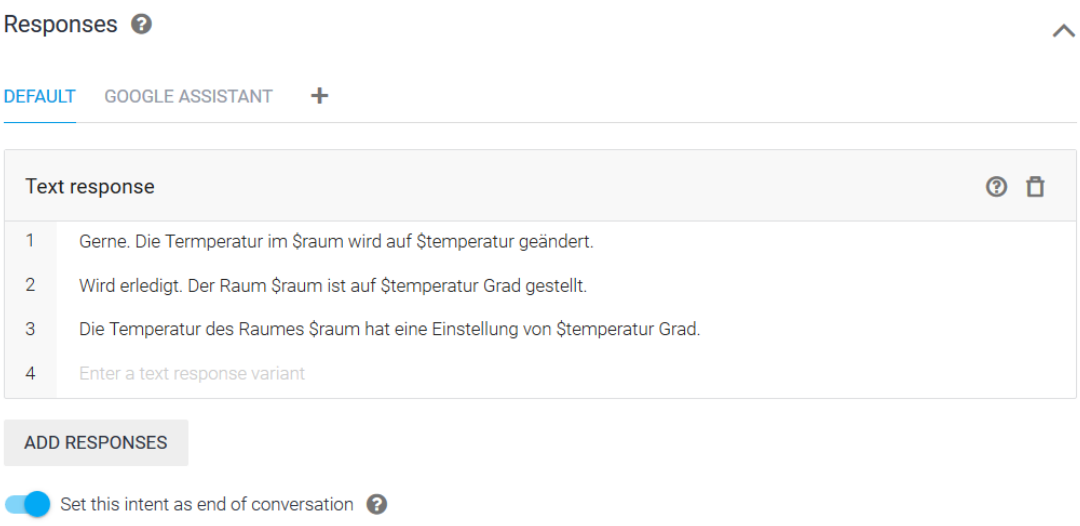

Abbildung 32: Antworten des Dialogmanagers auf den TemperaturIntent

A3 Quellcodeauszüge

A3.1 Grundlagen

A3.1.1 Manifest-Datei des Smart Home Systems

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="com.google.android.things.permission.MODIFY_SCREEN_SETTINGS" />
<uses-permission android:name="com.google.android.things.permission.MANAGE_INPUT_DRIVERS" />
<uses-permission android:name="com.google.android.things.permission.MANAGE_SENSOR_DRIVERS" />

<application>
    <uses-library android:name="com.google.android.things" />

    <activity
        android:name=".MainActivity"
        android:screenOrientation="reverseLandscape">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
            <category android:name="android.intent.category.IOT_LAUNCHER" />
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
    </activity>

    <activity
        android:name=".Sensor_BMP280.TemperatureSensorActivity"
        android:screenOrientation="reverseLandscape"
        tools:ignore="InnerclassSeparator">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

Abbildung 33: Manifest.xml des Smart Home Systems

A3.1.2 Gradle-Abhängigkeiten der eingesetzten Werkzeuge

```
//For Pocketsphinx Projects
implementation project(':aars')
implementation project(':models')
implementation 'com.android.support:appcompat' +
        '-v7:27.1.0'

//Cloud Speech-To-Text
api "io.grpc:grpc-okhttp:$grpcVersion"
api "io.grpc:grpc-stub:$grpcVersion"
api 'javax.annotation:javax.annotation-api:1.3.2'
api group: 'com.google.api.grpc', name: 'grpc-google-cloud-speech-v1', version: '0.1.17'

api('com.google.auth:google-auth-library-oauth2-http:0.7.0') {
    exclude module: 'httpclient'
}

//Firebase
//noinspection GradleCompatible
implementation 'com.google.firebase:firebase-core:15.0.2'
implementation 'com.google.firebase:firebase-messaging:15.0.2'
implementation 'com.google.firebase:firebase-database:15.0.1'

//Dialogflow
implementation 'ai.api:libai:1.6.11'
implementation 'ai.api:sdk:2.0.7@aar'
implementation 'com.google.code.gson:gson:2.8.2'
implementation 'commons-io:commons-io:20030203.000550'
implementation 'javax.inject:javax.inject:1'

//BMP280 Sensor
implementation 'com.google.android.things.contrib:driver-bmx280:0.5'
implementation('com.google.api-client:google-api-client-android:1.22.0') {
    exclude group: 'org.apache.httpcomponents'
}
implementation('com.google.apis:google-api-services-pubsub:v1-rev12-1.22.0') {
    exclude group: 'org.apache.httpcomponents'
}
```

Abbildung 34: Auszug der Gradle-Abhängigkeiten der eingesetzten Werkzeuge

A3.2 Funktion Wakeword

```
@Override
public void onBeginningOfSpeech() {
    Log.i(TAG, msg: "WakewordListener is active");
}

@Override
public void onPartialResult(Hypothesis hypothesis) {
    if (hypothesis == null) {
        return;
    }

    String text = hypothesis.getHypstr();
    if (text.equals(WAKEWORD)) {
        mSpeechRecognizer.stop();
    }
}

@Override
public void onResult(Hypothesis hypothesis) {
    if (hypothesis == null) {
        return;
    }

    String text = hypothesis.getHypstr();
    if (WAKEWORD.equals(text)) {
        mSphinxListener.onWakewordDetected();
    }
}

@Override
public void onEndOfSpeech() {
    if (!mSpeechRecognizer.getSearchName().equals(WAKEUP_SEARCH)) {
        mSpeechRecognizer.stop();
    }
}

@Override
public void onError(Exception e) {
    Log.e(TAG, msg: "Error regarding SpeechRecognizer");
}

@Override
public void onTimeout() {
    mSpeechRecognizer.stop();
}
```

Abbildung 35: Implmentierung des RecognitionListeners in der BasePocketSphinx-Klasse

A3.3 Funktion Spracherkennung

A3.3.1 Validierung AccessToken und Management gRPC-Kanal

```
@Override
protected void onPostExecute(AccessToken accessToken) {
    mAccessTokenTask = null;
    final ManagedChannel channel = new OkHttpChannelProvider()
            .builderForAddress(HOSTNAME, PORT)
            .nameResolverFactory(new DnsNameResolverProvider())
            .intercept(new GoogleCredentialsInterceptor(new GoogleCredentials(accessToken)
                    .createScoped(SCOPE)))
            .build();
    mSpeechApi = SpeechGrpc.newStub(channel);

    // Schedule access token refresh before it expires
    if (mHandler != null) {
        mHandler.postDelayed(mFetchAccessTokenRunnable,
                Math.max(accessToken.getExpirationTime().getTime() -
                        System.currentTimeMillis() -
                        ACCESS_TOKEN_FETCH_MARGIN, ACCESS_TOKEN_EXPIRATION_TOLERANCE));
    }
}
```

Abbildung 36: Validierung AccessToken und Management gRPC-Kanal[222]

A3.3.2 Instanziierung des VoiceRecorders in der MainActivity

```
private VoiceRecorder mVoiceRecorder;

private void startVoiceRecorder() {
    if (mVoiceRecorder != null) {
        mVoiceRecorder.stop();
    }
    mVoiceRecorder = new VoiceRecorder(mVoiceVoiceRecorderCallback);
    mVoiceRecorder.start();
}

private void stopVoiceRecorder() {
    if (mVoiceRecorder != null) {
        mVoiceRecorder.stop();
        mVoiceRecorder = null;
    }
}
```

Abbildung 37: Instanziierung, Starten und Stoppen des VoiceRecorders

[222] in Anlehnung an: Thumar (2017).

A3.3.3 Konfiguration Recognizer und Erstellung Request-Objekt

```
public void startRecognizing(int sampleRate) {
    if (mSpeechApi == null) {
        Log.w(TAG, msg: "API not ready. Ignoring the request.");
        return;
    }

    mRequestObserver = mSpeechApi.streamingRecognize(mResponseObserver);

    StreamingRecognitionConfig streamingConfig = StreamingRecognitionConfig.newBuilder()
            .setConfig(RecognitionConfig.newBuilder()
                    .setLanguageCode("de-DE")
                    .setEncoding(RecognitionConfig.AudioEncoding.LINEAR16)
                    .setSampleRateHertz(sampleRate)
                    .build()
            )
            .build();

    StreamingRecognizeRequest streamingRecognizeRequest =
            StreamingRecognizeRequest.newBuilder().setStreamingConfig(streamingConfig).build();
    mRequestObserver.onNext(streamingRecognizeRequest);
}
```

Abbildung 38: Konfiguration des Recognizers und Erstellung des Requests[223]

[223] in Anlehnung an: Thumar (2017).

A3.4 Anwendungsfall Sensordaten

A3.4.1 Temperatur- und Luftdruck-Listener der SensorActivity

```
// Callback when SensorManager delivers temperature or pressure data.
private SensorEventListener mTemperatureListener = new SensorEventListener() {
    @Override
    public void onSensorChanged(SensorEvent event) {
        mLastTemperature = event.values[0];
        Log.d(TAG, msg: "sensor changed: " + mLastTemperature);
        updateTemperature(mLastTemperature);
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
        Log.d(TAG, msg: "accuracy changed: " + accuracy);
    }
};

private SensorEventListener mPressureListener = new SensorEventListener() {
    @Override
    public void onSensorChanged(SensorEvent event) {
        mLastPressure = event.values[0];
        Log.d(TAG, msg: "sensor changed: " + mLastPressure);
        updateBarometer(mLastPressure);
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
        Log.d(TAG, msg: "accuracy changed: " + accuracy);
    }
};
```

Abbildung 39: Temperatur- und Luftdruck-Listener[224]

[224] in Anlehnung an: Android Things o. V. (2018).

A3.4.2 Konstruktor des PubSubPublishers

```
PubSubPublisher(Context context, String appname, String project, String topic,
                int credentialResourceId) throws IOException {
    mContext = context;
    mAppname = appname;
    mTopic = "projects/" + project + "/topics/" + topic;

    mHandlerThread = new HandlerThread( name: "pubsubPublisherThread");
    mHandlerThread.start();
    mHandler = new Handler(mHandlerThread.getLooper());

    InputStream jsonCredentials = mContext.getResources().openRawResource(credentialResourceId);
    final GoogleCredential credentials;
    try {
        credentials = GoogleCredential.fromStream(jsonCredentials).createScoped(
                Collections.singleton(PubsubScopes.PUBSUB));
    } finally {
        try {
            jsonCredentials.close();
        } catch (IOException e) {
            Log.e(TAG,  msg: "Error closing input stream", e);
        }
    }
    mHandler.post(() → {
            mHttpTransport = AndroidHttp.newCompatibleTransport();
            JsonFactory jsonFactory = JacksonFactory.getDefaultInstance();
            mPubsub = new Pubsub.Builder(mHttpTransport, jsonFactory, credentials)
                    .setApplicationName(mAppname).build();
    });
}
```

Abbildung 40: Autorisierung HTTP-Transport im Konstruktor des PubSubPublishers[225]

A3.4.3 Erstellung des Datenobjekts der Pub/Sub-Nachricht

```
    private JSONObject createMessagePayload(float temperature, float pressure)
            throws JSONException {
        JSONObject sensorData = new JSONObject();
        if (!Float.isNaN(temperature)) {
            sensorData.put( name: "temperature", String.valueOf(temperature));
        }
        if (!Float.isNaN(pressure)) {
            sensorData.put( name: "pressure", String.valueOf(pressure));
        }
        JSONObject messagePayload = new JSONObject();
        messagePayload.put( name: "deviceId", Build.DEVICE);
        messagePayload.put( name: "channel",  value: "pubsub");
        messagePayload.put( name: "timestamp", System.currentTimeMillis());
        if (sensorData.has( name: "temperature") || sensorData.has( name: "pressure")) {
            messagePayload.put( name: "data", sensorData);
        }
        return messagePayload;
    }
```

Abbildung 41: JSON-Objekt der Pub/Sub-Nachrichtenübertragung[226]

---

[225] in Anlehnung an: Android Things o. V. (2018).
[226] in Anlehnung an: Android Things o. V. (2018).

A3.5 Anwendungsfall Begleitanwendung

A3.5.1 Senden einer Notification

```
    NotificationCompat.Builder notificationBuilder = null;
    try {

        notificationBuilder = new NotificationCompat.Builder( context: this)
                .setSmallIcon(R.mipmap.ic_launcher)
                .setContentTitle(URLDecoder.decode(notificationData.getTitle(), enc: "UTF-8"))
                .setContentText(URLDecoder.decode(notificationData.getTextMessage(),
                        enc: "UTF-8"))
                .setAutoCancel(true)
                .setSound(RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION))
                .setContentIntent(pendingIntent);

    } catch (UnsupportedEncodingException e) {
        Log.e(TAG, msg: "Error while encoding data.");
        e.printStackTrace();
    }

    if (notificationBuilder != null) {
        NotificationManager notificationManager =
                (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
        notificationManager.notify(notificationData.getId(), notificationBuilder.build());
    }
```

Abbildung 42: Android NotificationBuilder zur Anzeige einer Firebase-Nachricht[227]

A3.5.2 RaumklimaPOJO

```
public class RaumklimaPOJO {

    private float temperatur;

    private String raum;

    public String getRaum() {
        return raum;
    }

    public void setRaum(String raum) {
        this.raum = raum;
    }

    public float getTemperatur() {
        return temperatur;
    }

    public void setTemperatur(float temperatur) {
        this.temperatur = temperatur;
    }
}
```

Abbildung 43: Realtime Database-Objekt RaumklimaPOJO

[227] in Anlehnung an: Carlos (2016).

A4 Datenträger

ERKLÄRUNG

“Ich versichere hiermit, dass ich die Arbeit selbstständig verfasst, keine anderen Quellen und Hilfsmittel als die angegebenen benutzt und die Stellen der Arbeit, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen sind, in jedem einzelnen Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe. Das Gleiche gilt auch für eingefügte Zeichnungen, Kartenskizzen und Darstellungen.”

Münster, den 18.05.2018

Ort, Datum **Unterschrift**